НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМ. ІГОРЯ СІКОРСЬКОГО”

Факультет Інформатики та обчислювальної техніки
(повне найменування інституту, факультету)

Обчислювальної техніки
(повна назва кафедри)

До захисту допущено
Завідувач кафедри

_____ _____
(підпис) (ініціали, прізвище)

“___”_____2019р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки **6.050103 « Програмна інженерія »** _____
(код і назва)

на тему: Система формування звітів
_____

Виконав (-ла): студент (-ка) __4__ курсу, групи __ІП-53__
(шифр групи)

Романюк Вікторія Віталіївна _____
(прізвище, ім’я, по батькові) (підпис)

Керівник асист. Регіда П. Г. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант нормо контроль д.т.н, проф. Сімоненко В.П. _____ _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМ. ІГОРЯ СІКОРСЬКОГО”

Факультет Інформатики та обчислювальної техніки
(повне найменування інституту, факультету

Обчислювальної техніки

Освітньо-кваліфікаційний рівень **бакалавр**

Напрям підготовки **6.050103 « Програмна інженерія »**

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ _____
(прізвище ініціали) (підпис)

“___”_____2019 р.

ЗАВДАННЯ

на бакалаврський дипломний проект студента

Романюк Вікторії Віталіївни
(прізвище, ім’я, по батькові)

1. Тема проекту (роботи) *Система формування звітів*
керівник проекту (роботи) асист. Регіда П. Г. ,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від “23”04 2019 року №1180-С
1. Термін здачі студентом закінченого проекту (роботи) *20 червня 2019*р.
3. Вихідні дані до проекту (роботи) технічна документація. теоретичні та статистичні дані з формування звітів
4. Зміст розрахунково-пояснювальної записки (перелік питань, які розробляються)
Опис та аналіз предметної області,аналіз наявних аналогів, дослідження методики формування звітів, конструювання архітектури та розробка системи формування звітів, тестування та впровадження системи
5. Перелік графічного матеріалу (з точним позначенням обов’язкових креслень)
структурна схема системи, узагальнена схема процесу формування звітів, блок-схема алгоритму роботи програми.

6. Консультанта проекту (робота), з вказівкою розділів роботи, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	Сімоненко В. П.		

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів дипломного проекту (роботи)	Строк виконання етапів проекту(роботи)	Примітки
1.	Затвердження теми роботи	10.12.2019-15.12.2019	
2.	Вивчення та аналіз завдання	15.12.2019-15.03.2019	
3.	Розробка архітектури та загальної структури системи	15.01 2019-25.01.2019	
4.	Розробка структур окремих підсистем	25.01.2019-5.02.2019	
5.	Програмна реалізація системи	5.02.2019-15.03.2019	
6.	Оформлення пояснювальної	15.03.2019-20.04.2019	
7.	Захист програмного продукту	25.04.2019	
8.	Передзахист	29.04.2019	
9.	Захист	20.05 2019	

Студент-дипломник _____
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

Пояснювальна записка дипломного проекту складається з чотирьох розділів, містить 23 рисунка, 14 таблиць, 5 додатків та 3 джерела – загалом 50 сторінок.

Об’єкт дослідження: автоматизовані системи формування звітів. Мета дипломного проекту: розробка системи формування звітів.

У першому розділі описано та проаналізовано предметну область, проведено аналіз існуючих успішних рішень. У другому розділі спроектовано архітектуру системи та основні процеси формування звітів. У третьому розділі наведені особливості реалізації системи та окремих її компонентів. У четвертому розділі описано процес розгортання системи та надається інструкція користувача.

ABSTRACT

The explanatory note of the diploma project consist of four sections, contains 23 figures, 14 tables, 5 annexes and 3 sources – total 50 pages.

The object of study: automated reporting system. The aim of the diploma project: automation of reports generation process.

In the first section the subject area is analyzed and described, provided a review of existing solutions in described area. The second section describes system architecture and main business processes of generating reports. The third section presents implementation details of the whole system and of each component. The fourth section describes deployment process and user manual.

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЕКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1.	А4		Завдання на дипломний проект	1	
2.	А4	ИАЛЦ.467200.001 ВП	Відомість проекту	1	
3.	А4	ИАЛЦ.467200.002 ТЗ	Технічне завдання	3	
4.	А4	ИАЛЦ.467200.003 ПЗ	Пояснювальна записка	50	
5.	А4	ИАЛЦ.467200.004 Д1	Схема алгоритму роботи програми	1	
6.	А3	ИАЛЦ.467200.005 Д2	Схема взаємодії класів системи	1	
7.	А3	ИАЛЦ.467200.006 Д3	Діаграма класів програми	1	
8.	А3	ИАЛЦ.467200.006 Д4	Схема бізнес процесів системи	1	
9.	А4	ИАЛЦ.467200.007 Б1	Лістинг програми	25	

					ІАЛЦ.467200.001 ВП
Ізмн.	Арк.	№ докум.	Підпис	Дата	
Розробн.		Романюк В. В.			*Система формування звітів*
Перевір.		Регіда П. Г.			
Реценз.					***Відомість дипломного проекту***
Н. Контр.		Сімоненко В.П.			
Затверд.		Стіренко С. Г.			

Літ.	Арк.	Аркушів
	1	1

НТУУ «КПІ ім. І. Сікорського»

ЗМІСТ

1. Найменування та область застосування

Дане технічне завдання поширюється на розробку системи формування звітів. Область застосування: інтернет-додаток “Система формування звітів”.

2. Підстави для розробки

Підставою для розробки системи електронно інвентаризації є завдання на дипломне проектування, затверджене кафедрою обчислювальної техніки Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (НТУУ «КПІ ім. І. Сікорського»)

3. Мета та призначення розробки

Метою даного проекту є розробка системи формування звітів. Призначення даної розробки полягає в створенні програмного забезпечення яке дозволить автоматизувати процес створення, зберігання та доповнення різноманітних наборів даних, а також формування звітів на основі зібраної інформації.

4. Джерела розробки

Джерелом розробки є науково-технічна література по інформаційним питанням та публікації в глобальній мережі Інтернет з даних питань.

5. Технічні вимоги

5.1. Вимоги до розроблюваного продукту

- можливість створення, редагування та видалення записів з наборів даних;
- можливість вибірки та фільтрації даних;

- можливість формування звітів на основі існуючих наборів даних, додавання різних типів графіків та діаграм до звітів;
- можливість імпорту готових наборів даних у систему (у форматах .csv, .xls);
- експорт наборів даних (.csv, .pdf, .xls);
- експорт сформованих звітів (.pdf).

5.2. Вимоги до програмного забезпечення

- операційна система MS Windows, Linux, MacOS;
- веб-браузер Google Chrome, Opera, Safari, Firefox, MS Edge, IE 9+.

5.3. Вимоги до апаратного забезпечення

- 32-розрядний (x86) або 64-розрядний (x64) процесор із тактовою частотою 1 ГГц або швидший;
- 1 гігабайт (ГБ) RAM;
- графічний пристрій із підтримкою DirectX 9.

6. Етапи розробки

Етап	Дата
1. Вивчення літератури за тематикою проекту	24.11.2018
2. Складання і узгодження технічного завдання	25.12.2018
3. Аналіз області застосування та особливостей системи	15.01.2019
4. Розробка загальної архітектури системи	15.03.2019
5. Реалізація програмного забезпечення	05.04.2019
6. Тестування програмного забезпечення	13.04.2019
7. Оформлення документації дипломної роботи	15.04.2019

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

BPMN – система умовних позначень (нотація) для моделювання бізнес-процесів.

БД – база даних.

UML – уніфікована мова моделювання.

Java EE – Java Platform, Enterprise Edition, обчислювальна корпоративна платформа Java.

JSF – JavaServer Faces, це каркас веб-застосунків написаний на Java.

HTML – Hypertext Markup Language, стандартна мова розмітки для створення веб-сторінок і веб-додатків.

CSS – Cascading Style Sheets, спеціальна мова, що використовується для опису зовнішнього вигляду сторінок, написаних мовами розмітки даних.

MVC – Model-view-controller, архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення.

ВСТУП

Звіт – це формальне представлення даних, що виводиться на екран, друкується або зберігається в файл для подальшого використання та обробки. Звіти дають можливість представити певний набір даних у зручному для сприйняття вигляді та допомагають у подальшому аналізі даних.

Створення професійних звітів потребує чимало часу та здібностей, так як потрібно зібрати всю необхідну інформацію з різних джерел та об’єднати її у єдиний формат у вигляді різноманітних графіків. Тому програмне забезпечення, яке здатне полегшити та автоматизувати процес створення звітів, набуло великої популярності. Дані рішення є необхідністю для представлення та проведення аналізу інформації, оскільки у сучасному світі обсяги даних є достатньо великими для ручної обробки.

Завданням даної роботи є розробка системи формування звітів, яка надасть можливість створення, зберігання та доповнення різноманітних наборів даних, а також формування звітів на основі зібраної інформації. Результатом роботи є інтегрована система, яку можна використовувати у будь-якій галузі діяльності, де необхідно автоматизувати процес накопичування інформації та створення звітів на її основі.

РОЗДІЛ 1
АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Для прийняття важливих рішень підприємцям необхідно бачити актуальну картину стану власного бізнесу. Для цих цілей підприємства користуються різноманітними видами звітів. Звітність підприємства використовується самими власниками, підприємствами для аналізу та контролю виконання договірних зобов'язань, аналізу господарської діяльності, для складання планів на майбутнє.

Користувачами звітів є наявні та потенційні інвестори, працівники, постачальники та інші торгові кредитори, замовники, уряд та урядові установи, громадськість, інші фізичні та юридичні особи. Вони використовують фінансові звіти для задоволення різноманітних потреб у фінансовій інформації. Такі потреби, зокрема, мають:

- Інвестори.
- Працівники, які зацікавлені в інформації щодо стабільності та прибутковості роботодавців.
- Постачальники та інші торгові кредитори, що зацікавлені в інформації чи будуть вчасно сплачені заборговані їм суми.
- Клієнти.
- Уряд та урядові установи.
- Громадськість – фінансові звіти можуть допомогти громадськості наданням інформації щодо останніх тенденцій і досягнень у соціальній сфері підприємства та обсягів його діяльності.

Процес створення звітів є доволі складним. Це зумовлено великим об'ємом інформації, на основі якої потрібно сформувати звіт. Найчастіше

необхідні дані знаходяться на різних ресурсах, відрізняються за структурою та форматом, що також додатково ускладнює формування звітів. Адже у цьому випадку спочатку необхідно зібрати всі дані та привести їх до єдиного вигляду та формату. Подібна робота може зайняти від декількох годин до декількох днів. Усе це зумовлює необхідність створення систем, що надають можливість полегшити та автоматизувати процес формування звітів.

1.2 Змістовний опис і аналіз предметної області

1.2.1 Класифікація звітів

З метою впорядкування складання звітності її класифікують за такими найбільш поширеними ознаками:

- Змістом і джерелами формування.
- Терміном подання.
- Ступенем узагальнення.
- Обсягом.
- Періодичністю подання.
- Охопленням видів діяльності.
- Поширенням на галузі народного господарства.
- Характером спрямування і використання.
- Ступенем використання обчислювальної техніки.

За змістом і джерелами формування звітність буває:

- Статистична.
- Фінансова.
- Податкова.
- Спеціальна.
- Внутрішньогосподарська.

Статистична звітність містить необхідну інформацію для статистичного вивчення господарської діяльності підприємств.

Фінансова звітність містить інформацію про фінансове становище, результати діяльності та рух грошових коштів підприємства за звітний період.

Податкова звітність містить інформацію про валові доходи та валові витрати, фінансові результати та розрахунок сум податків, що підлягають сплаті до бюджету, а також надмірно сплачених сум, що підлягають відшкодуванню.

Спеціальна звітність подається з питань розрахунків і використання коштів фонду соціального страхування, пенсійного фонду тощо.

Внутрішньогосподарська звітність відображає необхідну інформацію для прийняття рішень на рівні структурних підрозділів і розробляється підприємством самостійно.

За термінами подання розрізняють нормативну і строкову звітність. Нормативна подається на певну дату, строкова — у термін до 25 днів після закінчення звітного періоду.

За ступенем узагальнення звітність поділяють на первинну, що подається підприємствами, і зведену, що узагальнює дані первинної звітності у межах міністерств і відомств.

За обсягами відображених результатів діяльності розрізняють повну і скорочену звітність.

За періодичністю подання розрізняють річну і проміжну (щоквартальну, щомісячну) звітність.

З погляду охоплення видів діяльності звітність може відображати усі види діяльності, якою займається підприємство, або обмежуватися лише основним із них.

З погляду поширення на галузі народного господарства звітність є типовою і галузевою. Типові форми звітності застосовують для відображення облікових даних однакового змісту, галузеві містять показники за специфічними видами діяльності.

За характером спрямування розрізняють внутрішню звітність, призначену для внутрішнього управління підприємством, і зовнішню, яка виходить за межі підприємства і подається органам виконавчої влади, іншим користувачам.

Звітність за способом подання користувачам поділяють на подану поштовим зв'язком, телеграфом, електронною поштою або подану власноруч.

1.2.2 Технологія формування звітів

Першим етапом формування звіту є збір та систематизація усієї необхідної інформації. Також в рамках даного етапу проводиться уточнення інформації та виявлення і виправлення помилок. Дані дії є підготовчим етапом створення звіту.

Наступний етап – це планування та розробка моделі готового звіту. Потрібно провести аналіз зібраних даних, побудувати план представлення даних у готовому звіті, прийняти рішення щодо необхідності використання графіків та діаграм. Зовнішній вигляд та наповнення звіту у великій мірі залежить від аудиторії, для якої він призначається. Таким чином одні й ті самі дані фінансової звітності будуть представлені одним чином для урядових установ, та зовсім іншим чином для потенційних інвесторів або акціонерів. У першому випадку для урядових установ цінність мають саме ‘сухі’ дані, на основі яких можуть бути призначенні до сплати певні податки. Для інвесторів ж, навпаки, необхідно представити та візуалізувати дані з різних боків та показати результат аналізу різноманітних показників роботи та розвитку підприємства.

На останньому етапі потрібно на основі проведеного аналізу та розробленого плану побудувати тіло звіту. Для кращого сприйняття інформації аудиторією необхідно візуалізувати дані у вигляді різноманітних графіків та діаграм. Завершивши формування звіту потрібно зберегти його у зручному для подальшого застосування форматі та, за необхідністю, відправити на перевірку уповноваженим особам.

1.3 Аналіз успішних IT-проектів

Існує велика кількість онлайн-застосувань та програм для автоматизації процесу формування звітів. Більшість таких сервісів мають пробний безкоштовний період(або ознайомчу версію), а для доступу до всіх можливостей на постійний період необхідно придбати платний план користувача. Вони відрізняються кількістю наявних шаблонів, кількістю одночасних користувачів, обсягом наданого хмарного простору для зберігання файлів, розміром бази даних тощо.

Серед функцій даних застосувань представлені:

- Можливість створення шаблонних та персональних наборів даних.
- Вибірка існуючих даних за певний період часу з можливістю налаштування необхідних полів вибірки.
- Налаштування процесу формування звіту.
- Деякі з рішень надають можливість побудови графіків та діаграм.
- Експорт даних у декількох розповсюджених форматах (.csv, .xls, .pdf) .

1.3.1 Аналіз відомих програмних продуктів

Найбільш популярні платформи для формування звітів:

GetReport – найбільш відомий серед існуючих сервіс для автоматизування процесу збору інформації та побудови звітів.

Сервіс пропонує пробний безкоштовний період користування на 1 місяць. Для постійного користування існує декілька платних планів користувача: стартовий, базовий, професійний та корпоративний. Вони різняться кількістю користувачів, розміром сховища файлів та розміром бази даних.

Система є зручною у користуванні, має зручний графічний інтерфейс та можливість створювати персональні набори даних у вигляді таблиць, проводити вибірку та редагування даних, будувати графіки і діаграми та групувати їх по сторінкам, створювати звіти на основі існуючої інформації.

Mr.Doc – ця система є більш зручною для формування фінансової звітності та для створення звітів для роботи підприємств. Сервіс надає можливість створення аналітичних звітів по контролю замовлень покупців, замовлень постачальникам, цінових політик, залишків товарів на складі та графіки грошових коштів та продажів.

Система пропонує демо-акаунт для демонстрації можливостей платформи безкоштовно та без попередньої реєстрації. Для постійного корситування сервісом необхідно придбати платний план користувача, які різняться за доступною максимальною кількістю користувачів та наявністю певних додаткових можливостей.

Недоліком даної платформи є відсутність можливості побудови цілісних звітів, що складаються з декількох наборів даних та містять різноманітні графіки та діаграми. Сервіс надає можливість лише створення лише окремих вибірок даних та лише певних графіків. Користувачу необхідно скористатися додатковими рішеннями для отримання повного, готового звіту. Також експорт даних можливий лише у форматі .csv.

1.4 Аналіз вимог до програмного забезпечення

Для визначення вимог до програмного забезпечення необхідно визначити ролі користувачів та їх можливості в системі. Для досягнення мети розробки система повинна містити наступні типи користувачів:

– Адміністратор.
– Авторизований користувач.
– Автентифікований користувач.
– Неавтентифікований користувач.

Адміністратор має можливість створювати групи працівників та додавати користувачів до створених груп, можливість конструювання, редагування та видалення наборів даних, та має можливість вибірки та фільтрації даних.

Авторизований користувач має можливість переглядати, редагувати та доповнювати доступний йому існуючий набір даних, має можливість формувати графіки, проводити вибірку даних та експортувати дані.

Автентифікований користувач має доступ до збереження особистих данних для авторизації. Неавтентифікований користувач має можливість автентифікації.

Загалом система повинна мати такий функціонал:

– Автентифікація користувача.
– Реєстрація користувача.
– Конструювання наборів даних.
– Створення, редагування та видалення записів з наборів даних.
– Вибірка та фільтрація даних.
– Формування звітів на основі існуючих наборів даних, додавання різних типів графіків та діаграм до звітів.
– Імпорт готових наборів даних у систему (у форматах .csv, .xls).
– Експорт наборів даних (.csv, .pdf).

– Експорт сформованих звітів (.pdf).

1.4.1 Розроблення функціональних вимог

В системі передбачено наступні варіанти використання, що наведені у таблицях 1.1 – 1.6:

Таблиця 1.1 Варіант використання UC001

Назва	Авторизація адміністратора.
Опис	Адміністратор має можливість авторизуватись в системі.
Учасники	Адміністратор.
Передумови	
Постумови	Адміністратор проходить авторизацію у застосунку.
Основний сценарій	Система демонструє вікно авторизації, яке містить адрес електронної пошти та пароль. Адмінісратор заповнює поля вводу. Система демонструє кнопку “Увійти”. Адмінісратор натискає кнопку “Увійти”. Система авторизує адміністратора.

Продовження таблиці 1.1 Варіант використання UC001

Назва	Авторизація адміністратора.
Розширення сценаріїв	Система виявляє, що дані, внесені адміністратором, не є валідними. Система демонструє користувачеві повідомлення про відповідну помилку.

Таблиця 1.2 Варіант використання UC002

Назва	Експорт набору даних.
Опис	Користувач має можливість експортувати готовий набір даних.
Учасники	Користувач.
Передумови	Авторизований користувач.

Продовження таблиці 1.2 Варіант використання UC002

Назва	Експорт набору даних.
Постумови	Набір експортовано.
Основний сценарій	Користувач надсилає запит на перегляд доступних йому наборів даних. Система надає список доступних користувачеві наборів даних. Користувач обирає потрібний набір серед доступних. Система відображає обраний набір даних. Користувач обирає потрібний формат та натискає кнопку експорту. Система надає користувачу файл з набіром даних у потрібному форматі

Таблиця 1.3 Варіант використання UC003

Назва	Створення нового набору даних адміністратором
Опис	Адміністратор має можливість створити новий набір даних та надати можливість доступу до нього певним користувачам або групам користувачів
Учасники	Адміністратор.
Передумови	Користувач авторизований як адміністратор.
Постумови	Новий набір даних створено та збережено до БД.

Продовження таблиці 1.3 Варіант використання UC003

Назва	Створення нового набору даних адміністратором
Основний сценарій	Адміністратор надсилає запит на створення нового набору даних. Система надає форму для створення набору даних. Адміністратор заповнює форму, вводить назву набору даних, необхідні поля та їх типи, обирає користувачів або групи користувачів, які будуть мати доступ до перегляду та редагування даного набору. Адміністратор натискає кнопку “Завершити”. Система перевіряє введені адміністратором дані. Система створює новий набір даних та зберігає його до БД. Система відображає повідомлення про завершення.
Назва	Створення нового набору даних адміністратором
Розширення сценаріїв	Система виявляє, що дані, внесені адміністратором, не є валідними; Система демонструє користувачеві повідомлення про відповідну помилку.

Таблиця 1.4 Варіант використання UC004

Назва	Перегляд набору даних користувачем.
Опис	Користувач має можливість перегляду доступних йому наборів даних.
Учасники	Користувач

Продовження таблиці 1.4 Варіант використання UC004

Назва	Перегляд набору даних користувачем.
Передумови	Авторизований користувач
Постумови	Набір даних переглянуто.
Основний сценарій	Користувач надсилає запит на перегляд доступних йому наборів даних. Система надає список доступних користувачеві наборів даних. Користувач обирає потрібний набір серед доступних. Система відображає обраний набір даних у режимі перегляду. Користувач має можливість виконати фільтрацію, сортування та експорт набору даних. По закінченню роботи користувач закриває вікно з набором даних.

Таблиця 1.5 Варіант використання UC005

Назва	Редагування набору даних користувачем.
Опис	Користувач має можливість редагування доступних йому наборів даних.
Учасники	Користувач
Передумови	Авторизований користувач
Постумови	Набір даних відредаговано.

Продовження таблиці 1.5 Варіант використання UC005

Назва	Редагування набору даних користувачем.
Основний сценарій	Користувач надсилає запит на редагування доступних йому наборів даних. Система надає список доступних користувачеві наборів даних. Користувач обирає потрібний набір серед доступних. Система відображає обраний набір даних у режимі редагування. Користувач має можливість виконати редагування набору даних. По закінченню роботи користувач натискає кнопку “Зберегти зміни” та закриває вікно з набором даних. Система проводить перевірку введених даних. Система зберігає зміни до БД.

Таблиця 1.6 Варіант використання UC006

Назва	Побудова графіку користувачем.
Опис	Користувач має можливість створення графіків на основі доступних йому наборів даних.
Учасники	Користувач
Передумови	Авторизований користувач
Постумови	Графік побудовано.

Продовження таблиці 1.6 Варіант використання UC006

Назва	Побудова графіку користувачем.
Основний сценарій	Користувач надсилає запит на побудову графіку. Система відображає вікно створення графіку. Користувач обирає набір даних на основі якого буде побудовано графік. Користувач обирає колонки, по яким буде побудовано графік. Користувач обирає тип необхідного йому графіку. Користувач натискає на кнопку «Побудувати графік» Система проводить перевірку даних. Система будує графік та відображає його на екрані. Користувач за бажанням завантажує графік у форматі .pdf. По закінченню роботи користувач закриває вікно.

ВИСНОВКИ ДО РОЗДІЛУ 1

Формування та презентація звітності є важливою частиною аналізу діяльності підприємства, ефективності обраної стратегії розвитку тощо. У сучасних умовах для формування звітів доцільно використовувати автоматизовані системи, що здатні спростити та удосконалити даний процес. Існує ряд готових рішень для створення персональних наборів даних та формування звітів, вони мають ряд переваг та недоліків. Більшість існуючих рішень надають можливість провести аналіз лише одного набору даних за раз, що в результаті ускладнює процес формування готового звіту. Також не всі сервіси підтримують можливість імпорту готових наборів даних.

Розробка системи, яка б підтримувала описану вище функціональність, є актуальним рішенням серед аналогів.

Готова система повинна надавати можливість створення персональних наборів даних, додавання, редагування та видалення записів з наборів даних, формування звітів з різноманітними графіками та діаграмами, імпорту готових наборів даних у систему, експорту наборів даних у найбільш популярних форматах, експорту готових сформованих звітів у форматі .pdf.

РОЗДІЛ 2
МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Для створення якісного програмного забезпечення необхідно попереднє детальне моделювання процесів і архітектури програмного забезпечення. Для проектування бізнес-процесів було обрано методологію створення діаграм BPMN. У додатку А відображено схему процесів діяльності системи.

Схема містить процеси, які проходить користувач під час роботи з системою: аутентифікацію, реєстрацію, роботу з наборами даних та звітами.

2.1.1 Реєстрація та аутентифікація користувача

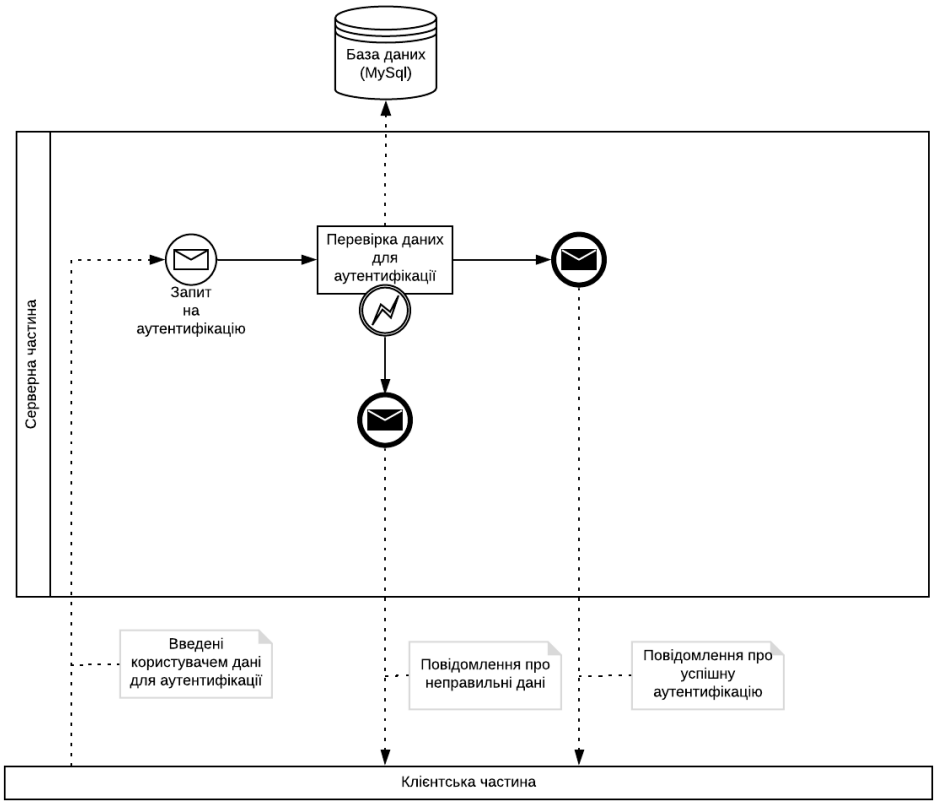


Рис. 0.1 Схема аутентифікаціїї користувача

Послідовний опис процесу аутентифікації користувача зображеного на рис. 2.1:

– Система демонструє панель аутентифікації користувача.
– Користувач вводить дані для аутентифікації. Відповіді користувача відправляються на сервер.
– Система виконує пошук користувача за введеними даними. Якщо дані введені вірно і користувача знайдено у базі даних, користувач успішно аутентифікований і має можливість перейти до подальшої роботи з системою.

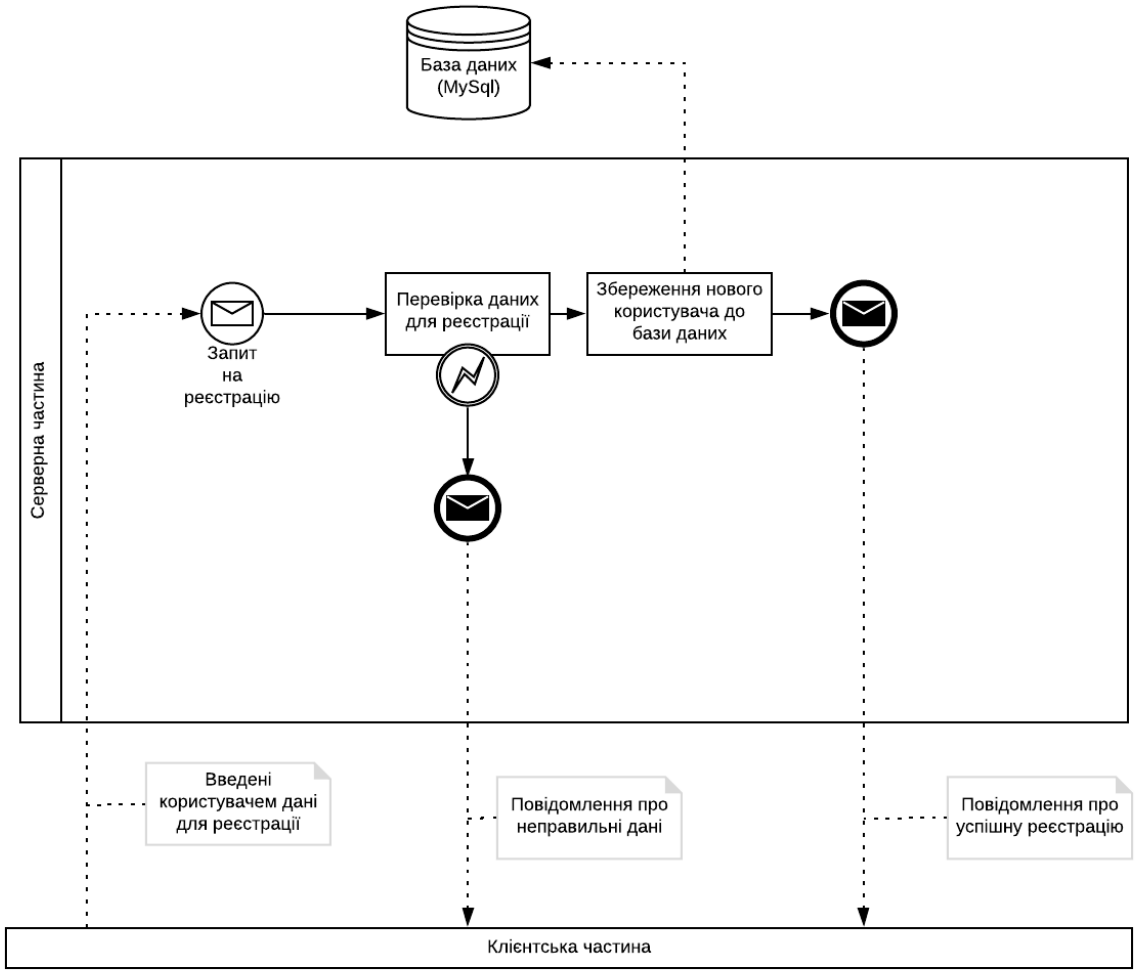


Рисунок 0.2 Схема реєстрації користувача

Послідовний опис процесу реєстрації користувача зображеного на рисунку 2.2 :

– Ссистема демонструє панель реєстрації користувача.
– Користувач вводить дані для реєстрації. Відповіді користувача відправляються на сервер.
– Система виконує перевірку введених даних. У разі успішної перевірки даних система зберігає нового користувача у базі даних та користувач перенаправляється на сторінку аутентифікації. У разі помилки система відправляє повідомлення про помилку.

2.1.2 Робота з наборами даних

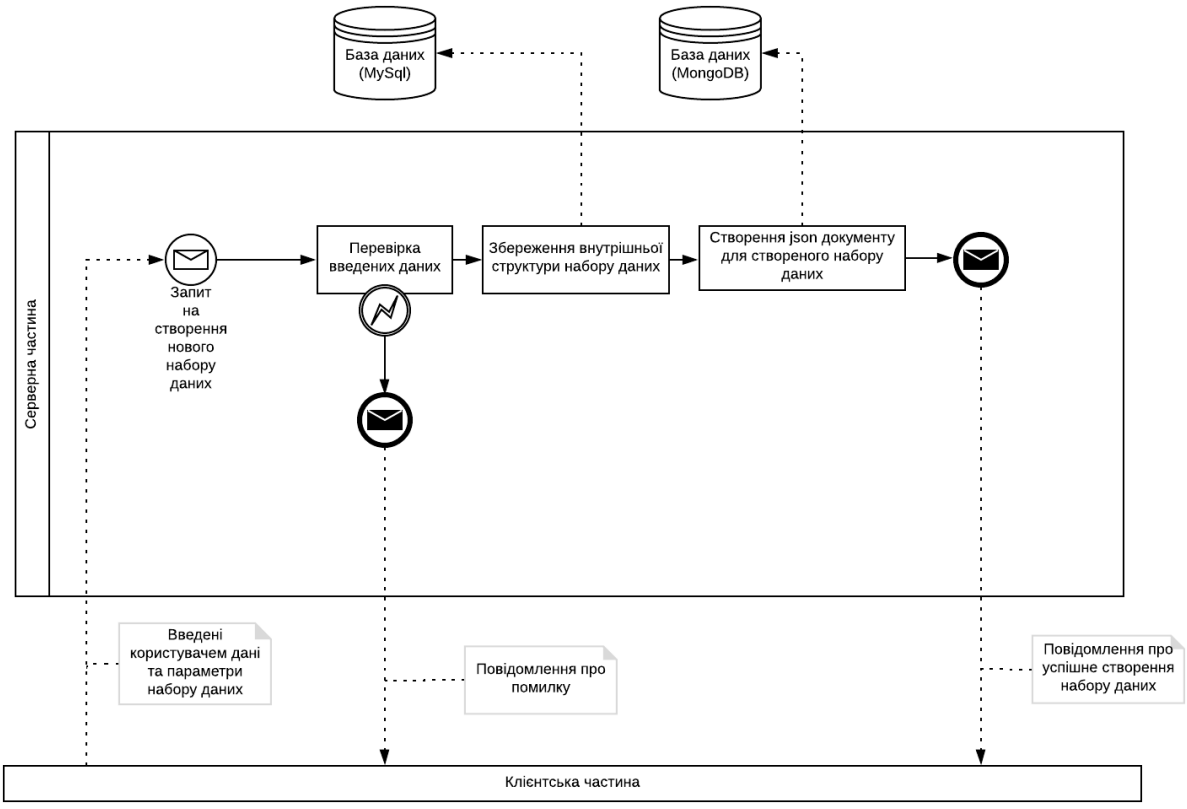


Рис. 0.3 Схема створення нового набору даних

Послідовний опис процесу створення нового набору даних користувачем зображеного на рис. 2.3:

– Система демонструє панель створення набору даних.
– користувач вводить необхідні дані для створення набору даних.
– система перевіряє введені дані.
– у випадку помилки демонструє повідомлення про помилку.
– у випадку правильно введених даних система зберігає струкуту нового набору даних до реляційної бази даних, створює відповідну json-версію набору, зберігає її до бази даних MongoDB та демонструє повідомлення про успішне створення набору даних.

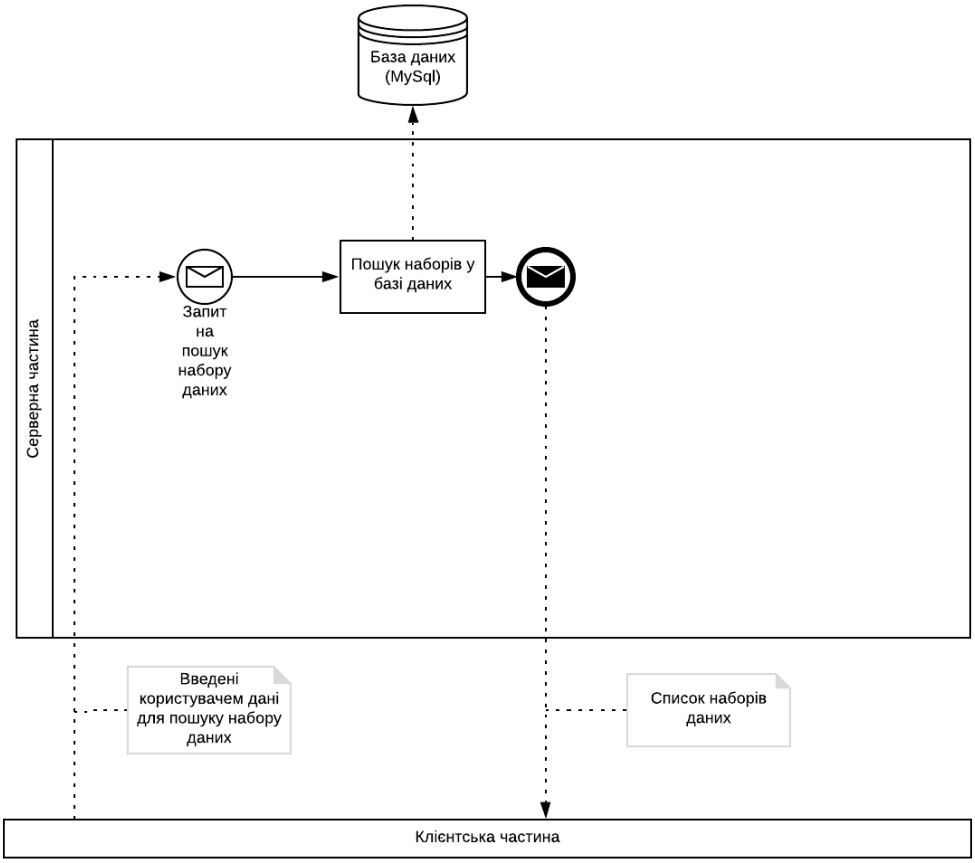


Рис. 0.4 Схема пошуку набору даних

Послідовний опис процесу пошуку набору даних користувачем зображеного на рис. 2.4:

– Система демонструє панель пошуку набору даних.
– Користувач вводить ім’я або частину імені набору для пошуку.
– Система виконую пошук відповідних наборів у базі даних.
– Система демонструє результати пошуку.

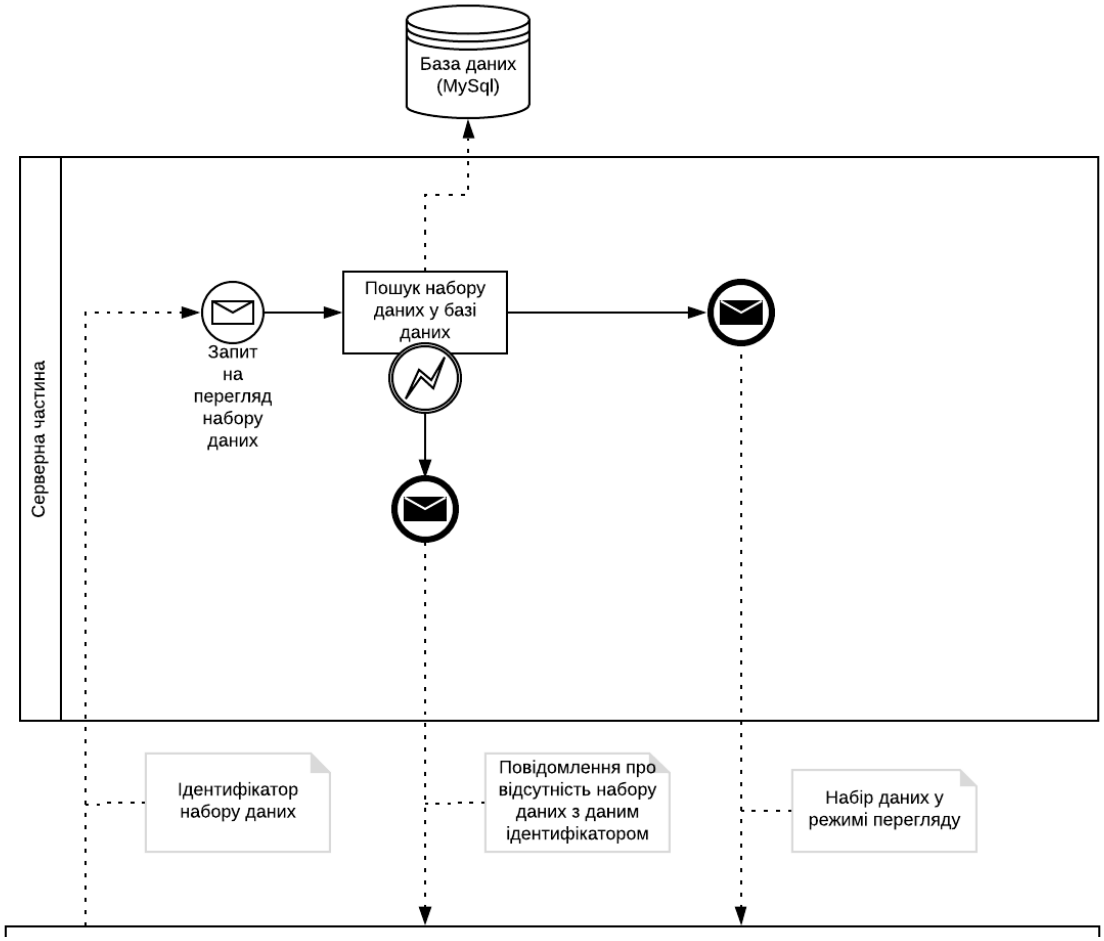


Рис. 0.5 Схема перегляду набору даних

Послідовний опис процесу перегляду набору даних користувачем зображеного на рис. 2.5:

– Користувач відправляє ідентифікатор потрібного набору даних та запит на перегляд набору.
– Система виконує пошук набору у базі даних.

– У разі відсутності набору з даним ідентифікатором система демонструє повідомлення про помилку.
– У разі успішного пошуку система надає набір даних користувачеві у режимі перегляду.

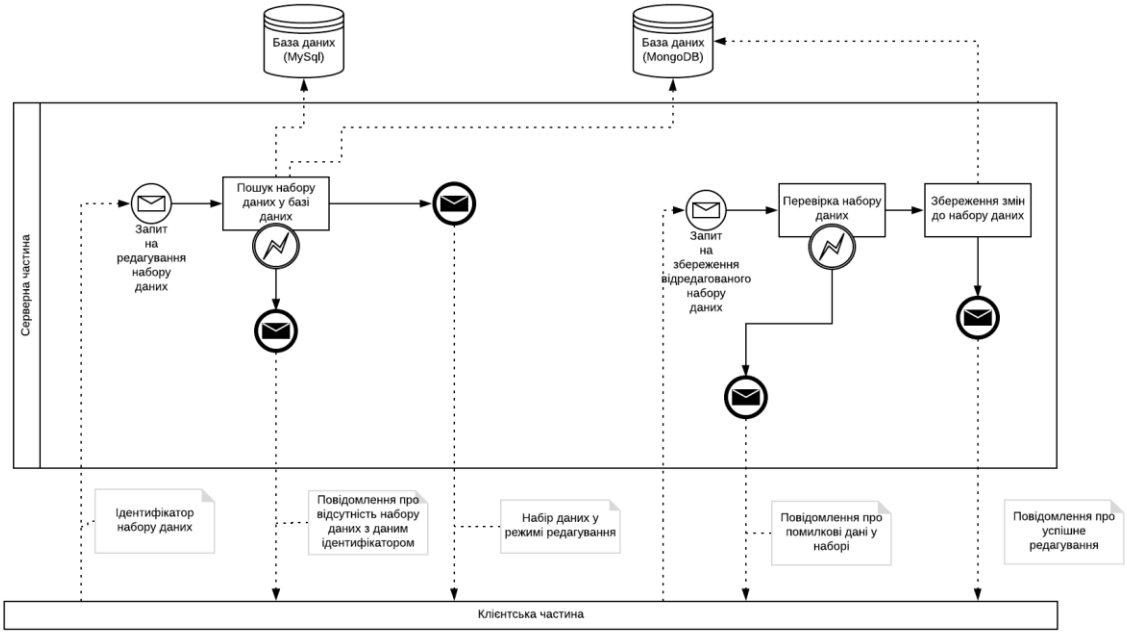


Рис. 0.6 Схема редагування набору даних

Послідовний опис процесу редагуваня набору даних користувачем зображеного на рис. 2.6:

– Користувач відправляє ідентифікатор потрібного набору даних та запит на перегляд набору.
– Система виконує пошук набору у базі даних.
– У разі відсутності набору з даним ідентифікатором система демонструє повідомлення про помилку.
– У разі успішного пошуку система надає набір даних користувачеві у режимі редагування.
– Користувач проводить необхідні маніпуляції з набором даних та натискає кнопку ‘Зберегти зміни’.

– Система проводить перевірку внесених змін.
– У разі помилок система демонструє повідомлення про помилку.
– У разі відсутності помилок система зберігає зміни до бази даних та демонструє повідомлення про успішне редагування набору.

2.1.3 Робота з графіками

Послідовний опис процесу створення нового графіку користувачем:

– Користувач відправляє запит на створення нового графіку.
– Система демонструє панель створення графіку.
– Користувач виконує конструювання графіку, використовуючи усі інструменти системи та натискає кнопку ‘Сформувати графік.
– Система перевіряє наявність усіх необхідних даних.
– У разі помилки система демонструє повідомлення про помилку.
– У разі відсутності помилок система проводить аналіз даних та формує графік.
– Користувач має можливість завантажити графік за у форматі .pdf.

2.2 Конструювання програмного забезпечення

Для вирішення поставленого завдання було обрано клієнт-серверну архітектуру. Архітектура клієнт-сервер є одним із архітектурних шаблонів програмного забезпечення та є домінуючою концепцією у створенні розподілених мережних застосунків і передбачає взаємодію та обмін даними між ними.

Вона передбачає такі основні компоненти:

– Набір серверів, які надають інформацію або інші послуги програмам, які звертаються до них.

– Набір клієнтів, які використовують сервіси, що надаються серверами.
– Мережа, яка забезпечує взаємодію між клієнтами та серверами.

Сервери є незалежними один від одного. Клієнти також функціонують паралельно і незалежно один від одного. Немає жорсткої прив'язки клієнтів до серверів. Більш ніж типовою є ситуація, коли один сервер одночасно обробляє запити від різних клієнтів; з іншого боку, клієнт може звертатися то до одного сервера, то до іншого. Клієнти мають знати про доступні сервери, але можуть не мати жодного уявлення про існування інших клієнтів.

Для розробки серверної частини було обрано архітектурний шаблон MVC, що передбачає поділ системи на три взаємопов'язані частини: модель даних, вигляд (інтерфейс користувача) та модуль керування. Застосовується для відокремлення даних (моделі) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача.

Мета шаблону — гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми. Крім того використання цього шаблону у великих системах сприяє впорядкованості їхньої структури і робить їх більш зрозумілими за рахунок зменшення складності.

На рисунку 2.3 зображена загальна схема архітектури системи.

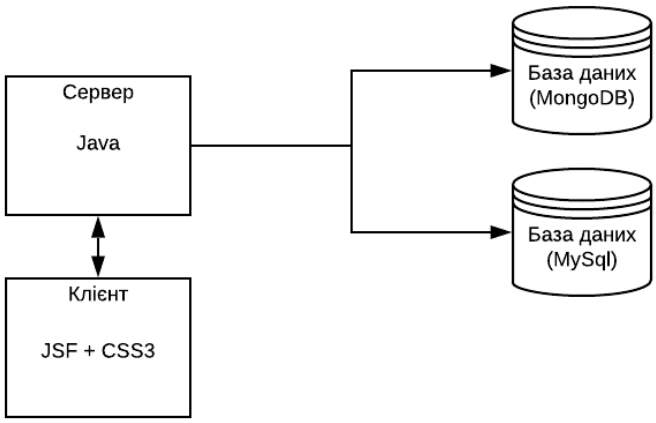


Рис. 0.3 Схема загальної архітектури системи

2.3 Програмне забезпечення системи

Архітектура системи побудована на таких складових:

– Серверна частина – Spring Boot, JPA(Hibernate), Spring Data, Spring Security.

– Клієнтська частина – JSF, CSS3, Bootstrap 3.

JavaServer Faces (JSF) — це каркас веб-застосунків написаний на Java. Він служить для того, щоб полегшувати розробку користувацьких інтерфейсів для Java EE застосунків. На відміну від більшості MVC фреймворків, які керуються запитами, підхід JSF ґрунтується на використанні компонентів. Стан компонентів користувацького інтерфейсу зберігається, коли користувач запитує нову сторінку й потім відновлюється, якщо запит повторюється. Для відображення даних звичайно використовується JSP, але JSF можна пристосувати й під інші технології, наприклад XUL.

Переваги:

– Генерація серверної частини інтерфейсу користувача.

– Базується на компонентах.
– Наявна обробка подій та станів.
– Різноманітні view-технології — не тільки HTML та JavaScript.
– Розробка з урахуванням доступного інструментарію.
– Наявний стандарт і він включений до Java EE.
– Рольова модель розробки.

Недоліки:

– Потрібно багато часу для вивчення та освоєння технології.
– Потребуються потужні обчислювальні можливості серверу.

Java Persistence API (JPA) — специфікація API Java EE, надає можливість зберігати в зручному вигляді Java-об’єкти в базі даних.

Існує декілька реалізацій даного інтерфейсу, одна з найбільш популярних – Hibernate. JPA реалізує концепцію ORM. ORM – технологія програмування, яка зв'язує бази даних з концепціями об'єктно-орієнтованих мов програмування, створюючи «віртуальну об'єктну базу даних».

ВИСНОВКИ ДО РОЗДІЛУ 2

У даному розділі було проаналізовано усі необхідні процеси системи формування звітів та створено відповідні схеми з використанням діаграм BPMN. Було розроблено архітектуру системи на базі клієнт-серверної моделі.

РОЗДІЛ 3
КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБАЗПЕЧЕННЯ

3.1 Програмні модулі

Основні програмні модулі системи наведені у таблиці 3.1.

Таблиця 3.1 Програмні модулі системи

№ з/п	Позначення	Призначення
1	HomeController.java	Контролер домашньої сторінки системи
2	AdminController.java	Контролер який відповідає за функціонал адміністративної панелі
3	UserController.java	Контролер який відповідає реєстрацію та аутентифікацію користувачів
4	CreateDatasetBean.java	Контролер який відповідає за створення нового набору даних
5	EditDatasetBean.java	Контролер який відповідає за редагування набору даних з можливістю експорту у формати .xls, .csv та .pdf
6	ViewDatasetBean.java	Контролер який відповідає за перегляд набору даних з можливістю експорту у формати .xls, .csv та .pdf
7	SearchDatasetBean.java	Контролер який відповідає за пошук наборів даних за ім’ям набору
8	CreateChartBean.java	Контролер який відповідає за формування графіків

Продовження таблиці 3.1 Програмні модулі системи

№ з/п	Позначення	Призначення
9	User.java	Модель що відповідає за таблицю користувачів у базі даних
10	Role.java	Модель що відповідає за таблицю типів користувачів у базі даних
11	Dataset.java	Модель що відповідає за таблицю набору даних у базі даних
12	DatasetColumn.java	Модель що відповідає за таблицю полів набору даних у базі даних
13	DatasetJson.java	Модель що відповідає за представлення набору даних у форматі json
14	DatasetJsonMaper.java	Сервіс для конвертування набору даних у формат json.
15	createDataset.xhtml	Сторінка створення нового набору даних
16	searchDataset.xhtml	Сторінка пошуку набору даних
17	viewDataset.xhtml	Сторінка перегляду набору даних
18	editDataset.xhtml	Сторінка редагування набору даних
19	importDataset.xhtml	Сторінка імпорту готового набору даних до системи
20	index.xhtml	Головна сторінка системи
21	login.html	Сторінка аутентифікації користувача
22	registration.xhtml	Сторінка реєстрації користувача

3.1.1 Опис методів інтерфейсів та класів

Далі опишемо основні методи інтерфейсів та класів застосунку (таблиці 3.2 – 3.7).

Таблиця 0.2 Опис методів класу CreateDatasetBean

Назва метода	Аргументи	Значення, що повертається	Опис
onAddNew()	–	–	Додає нову колонку до набору даних
deleteColumn()	Long id	–	Видаляє колонку за набору даних
isEmptyColumns()	–	boolean	Перевіряє чи є колонки в наборі даних
saveDataset()	Dataset	–	Зберігає набір даних до бази даних

Таблиця 0.3 Опис методів класу EditDatasetBean

Назва метода	Аргументи	Значення, що повертається	Опис
onAddNew()	–	–	Додає новий запис до набору даних
deleteRow()	RowWrapper selectedRow	–	Видаляє запис з набору даних
getDataset()	–	Dataset[]	Читає набір даних з бази даних

Продовження таблиці 0.3 Опис методів класу EditDatasetBean

Назва метода	Аргументи	Значення, що повертається	Опис
saveChanges()	–	–	Зберігає відредагований набір даних до БД

Таблиця 0.4 Опис методів класу SearchDatasetBean

Назва метода	Аргументи	Значення, що повертається	Опис
search()	String datasetName	Dataset[]	Виконує пошук набору даних в БД
isEmptyResult()	–	boolean	Перевіряє чи знайдено набір даних
getResultDatasets()	–	Dataset[]	Повертає знайдені набори даних

Таблиця 0.5 Опис методів класу UserController

Назва метода	Аргументи	Значення, що повертається	Опис
login()	User user	–	Аутентифікує користувача
register()	User user	–	Реєструє нового користувача

Таблиця 0.6 Опис методів класу DatasetJsonMapper

Назва метода	Аргументи	Значення, що повертається	Опис
mapToJson()	Dataset dataset, String[][] values	Dataset	Конвертує набір даних у формат json

Таблиця 0.7 Опис методів класу CreateChartBean

Назва метода	Аргументи	Значення, що повертається	Опис
generateChart()	ChartTypeEnum type	–	Створює графік типу type
createLineChart()	–	–	Створює лінійний графік
createPieChart()	–	–	Створює графік пиріг
searchDataset()	String datasetName	Dataset[]	Виконує пошук набору даних в БД
clean()	–	–	Очищає параметри створення графіку

Далі опишемо використані у системі для взаємодії між компонентами типи даних та структури даних у таблиці 0.2.

Таблиця 0.2 Типи даних та структури даних

Назва типу	Тип або структура даних	Опис
ChartTypeEnum	«LINE» | «PIE»	Тип графіку: LINE – лінійний, PIE – пиріг

Продовження таблиці 0.3 Типи даних та структури даних

Назва типу	Тип або структура даних	Опис
ColumnTypeEnum	«TEXT_TYPE» «NUMBER_TYPE» «DOUBLE_TYPE»	Тип значень колонки в наборі даних: TEXT_TYPE – текст, NUMBER_TYPE – цілі числа DOUBLE_TYPE – числа з плаваючою комою
Dataset	{ id: «long», name: «string», user: «User», columns: «DatasetColumn[]» }	Повний опис набору дагих у системі, що включає унікальний ідентифікатор, назву, користувача, якому належить набір та список колонок
DatasetColumn	{ id: «long», name: «string», type: «ColumnTypeEnum», dataset: «Dataset»}	Опис колонки набору даних у системі. Містить унікальний ідентифікатор, ідентифікатор набору даних, ім'я, та тип допустимих значень

Продовження таблиці 0.4 Типи даних та структури даних

Назва типу	Тип або структура даних	Опис
DatasetJson	{ id: «ObjectId», datasetId: «long», values: «String[][]»}	Опис набору даних у форматі .json. Містить унікальний ідентифікатор, ідентифікатор набору даних та масив значень набору даних.
User	{ id: «long», username: «string», password: «string», email: «string», roles: «Role[]»}	Опис користувача системи. Містить унікальний ідентифікатор, ім’я користувача, пароль та список ролей у системі.

ВИСНОВКИ ДО РОЗДІЛУ 3

У даному розділі було виконано конструювання програмного забезпечення, описано основні компоненти системи формування звітів та їх взаємодію та наведено деталі реалізації.

РОЗДІЛ 4
ВПРОВАДЖЕННЯ ТА ОГЛЯД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Для розгортання серверної частини програмного забезпечення, необхідною є наявність JDK, що можуть бути встановлена з офіційних сайтів для необхідної ОС.

4.2 Огляд інтерфейсу системи

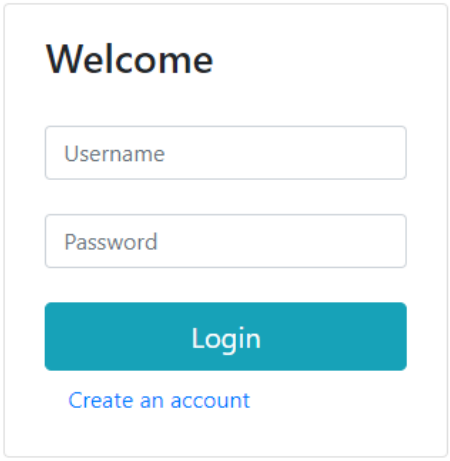


Рис. 4.1 Панель входу до системи

При початковому доступі до системи відображається вікно аутентифікації користувача зображене на рис. 4.1. Користувач має можливість надати дані для входу до системи.

У разі надання невірних даних або відсутності користувача у базі даних система відобразить повідомлення про помилку зображене на рис. 4.2.

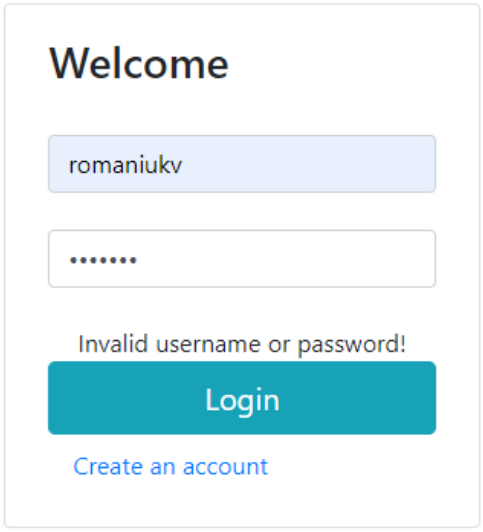


Рис 4.2 Приклад невырного вводу даних

У разі успішної аутентифікації користувач перенаправляється на головну сторінку системи.

У випадку коли користувач не зареєстрований у системі, він може перейти за посиланням до сторінки реєстрації нового користувача, що зображена на рис. 4.3.

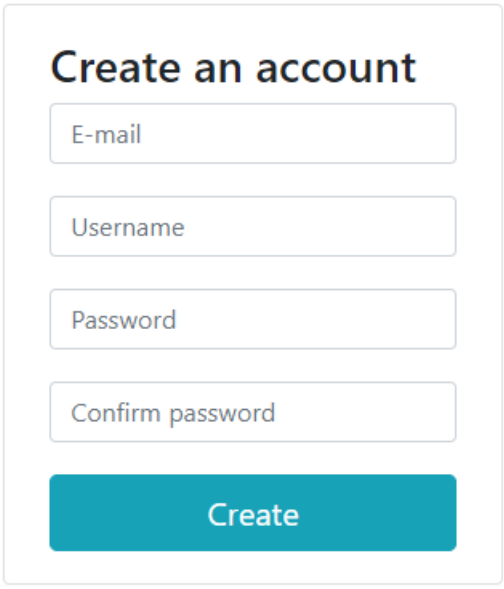


Рис. 4.3 Панель реєстрації

У разі успішної реєстрації користувач перенаправляється на сторінку аутентифікації.

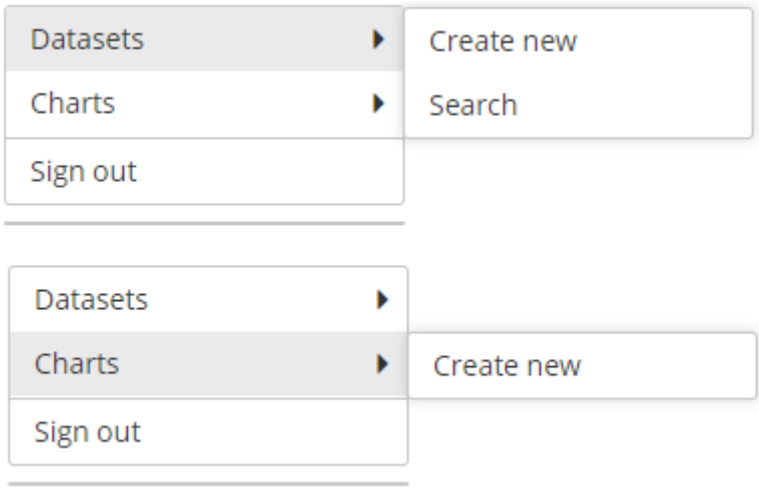


Рис. 4.4 Головне меню системи

На рис. 4.4 зображено головне меню системи та підпункти головного меню.

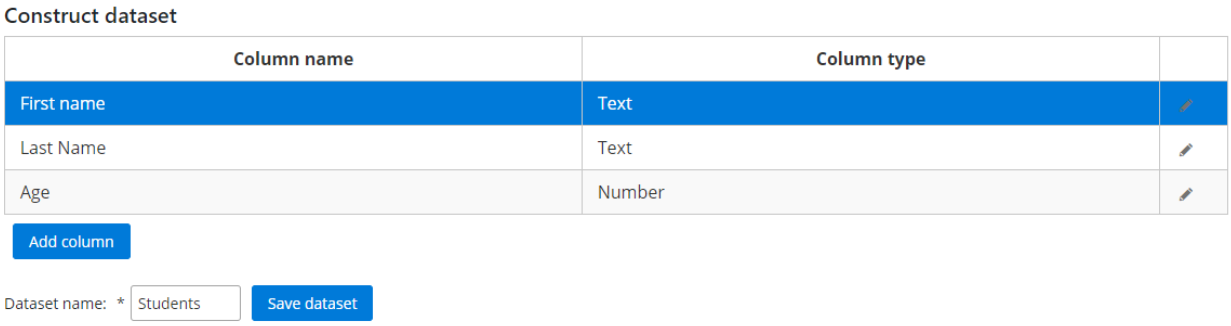


Рис. 4.5 Панель створення набору даних

На рисунку 4.4 зображено панель створення нового набору даних. Наявні три можливі типи даних – текст, ціле число та число з плаваючою комою. Видалення одного з полів відбувається натисканням правої кнопки миші та варіанту “Delete” (рис 4.5).

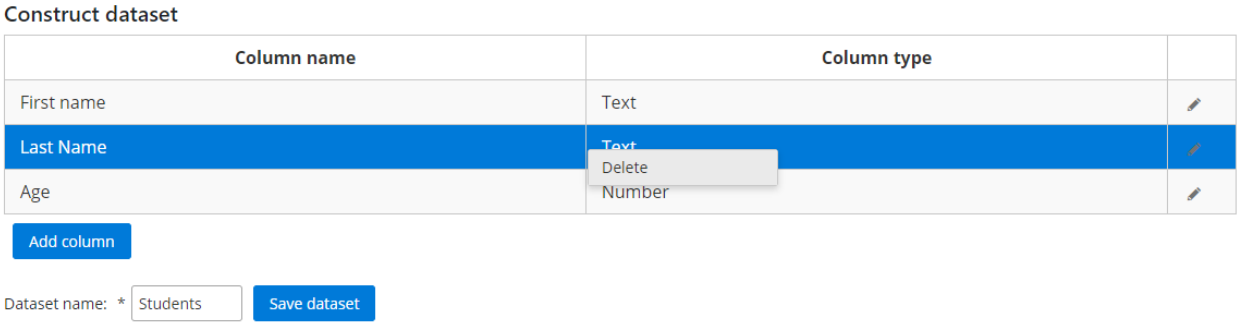


Рис. 4.6

Після натискання на кнопку “Save dataset” набір даних буде збережено до бази даних і система продемонструє повідомлення про успішне створення набору даних (рис. 4.6).

Рис. 4.7 Панель пошуку набору даних

На рис. 4.7 зображено панель пошуку набору даних. Наявна можливість перегляду та редагування знайдених наборів.

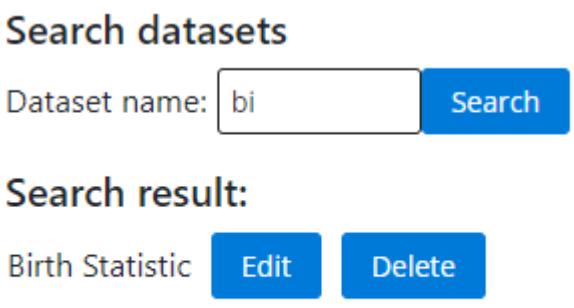


Рис. 4.8 Панель результату пошуку набору даних

На рис. 4.8 зображено панель результатів пошуку. Користувач має можливість переглянути знайдені набори або ж відредагувати та доповнити їх.

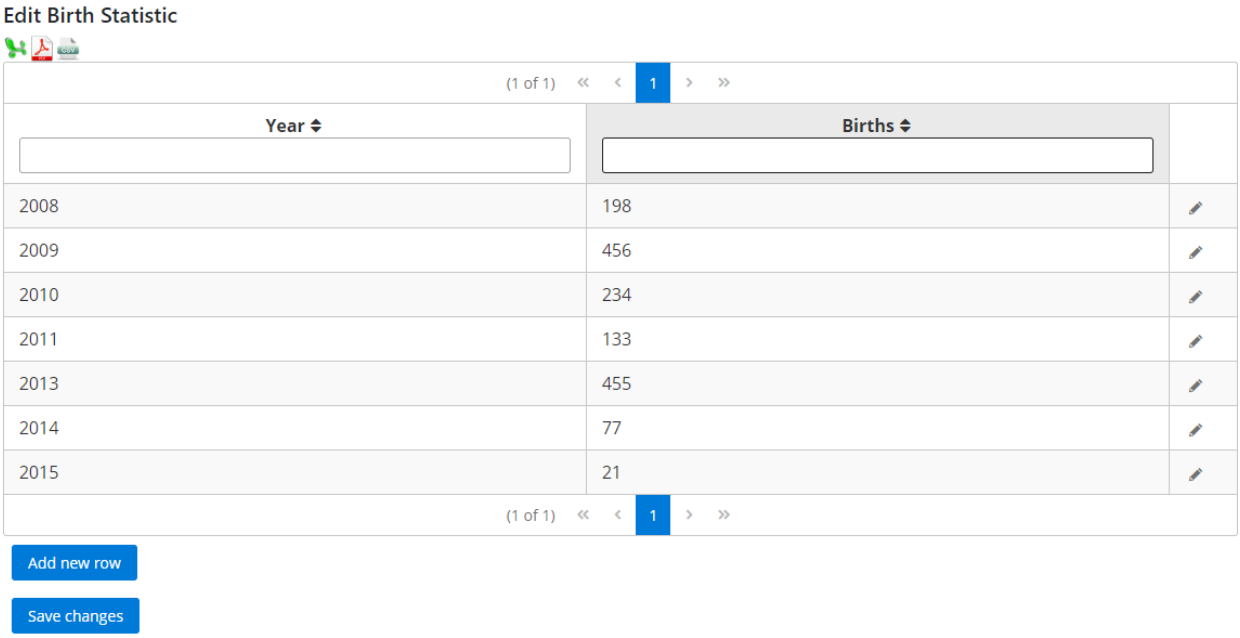


Рис. 4.9 Панель перегляду набору даних у режимі редагування

На рис. 4.9 наведено панель перегляду набору даних у режимі редагування. Користувач має можливість додати новий запис, видалити та внести зміни до існуючого запису. Також наявна можливість експорту набору набору даних за допомогою кнопок, зображених на рис. 4.10

Рис. 4.10 Іконки експорту набору даних

Рис. 4.11 Панель створення графіку – пошук набору даних

На рис. 4.11 зображено панель створення графіку. Користувач має можливість ввести ім’я набору даних, на основі якого буде створено графік. Після натискання кнопки “Search” система виконає пошук та надасть результати пошуку користувачеві (рис. 4.12)

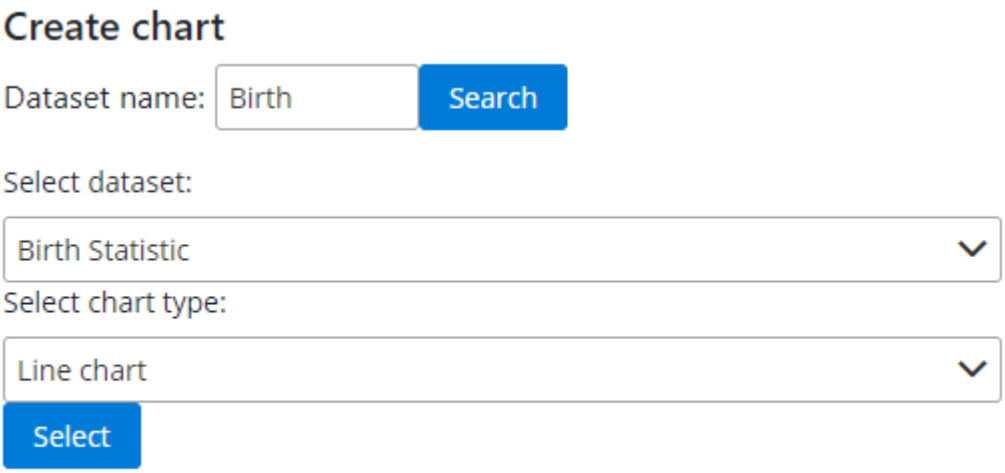


Рис. 4.12 Панель створення графіку – результати пошуку

Далі користувачу необхідно обрати потрібний набір даних та тип графіку.

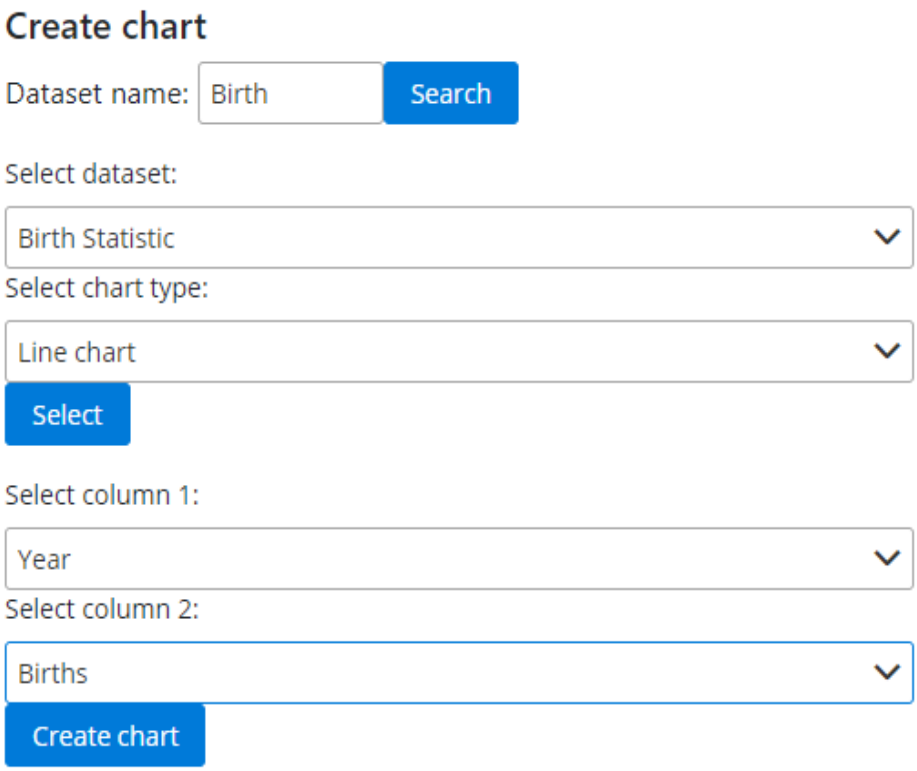


Рис. 4.13 Панель створення графіку – вибір колонок

Після натискання кнопки “Select” користувачу надається можливість обрати колонки по яким буде побудовано графік (рис. 4.13). По натисканню кнопки “Create chart” система проведе перевірку обраних даних та побудує графік. На рис. 4.14 наведено успішно побудований лінійний графік, на рис. 4.15 – графік пиріг.

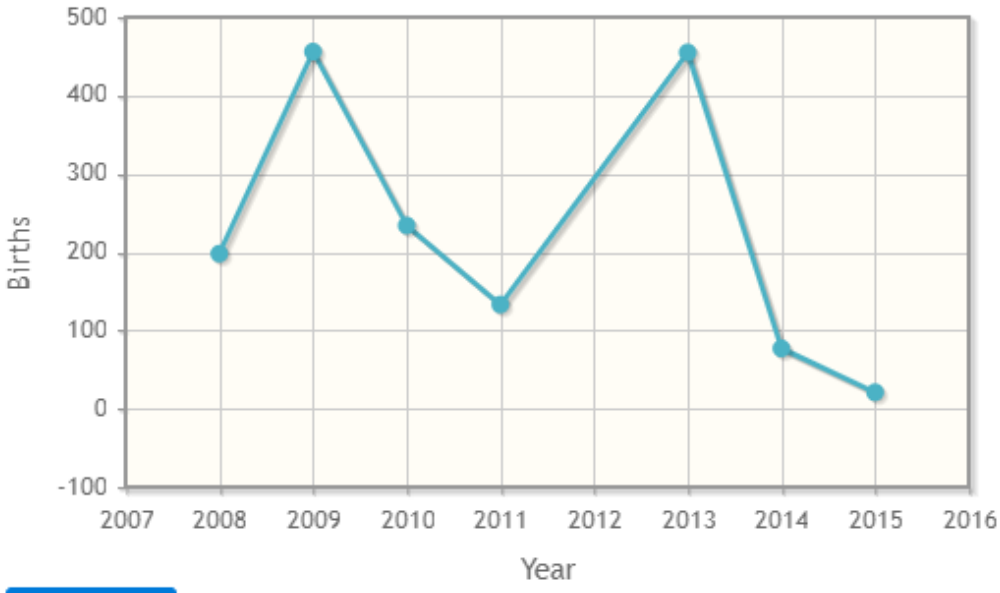


Рис. 4.14 Приклад лінійного графіку

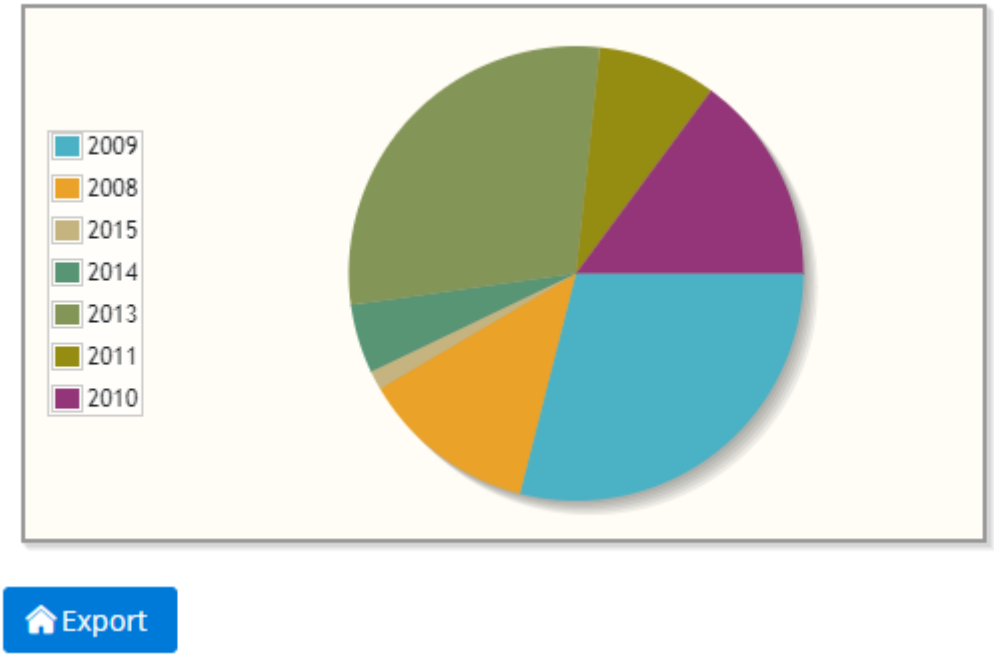


Рис. 4.15 Приклад графіку пиріг

У разі успішної побудови графіку користувач має можливість експортувати графік у форматі .pdf натиснуваши на кнопку “Export”. У разі помилки система відобразить повідомлення про помилку – рис. 4.16.

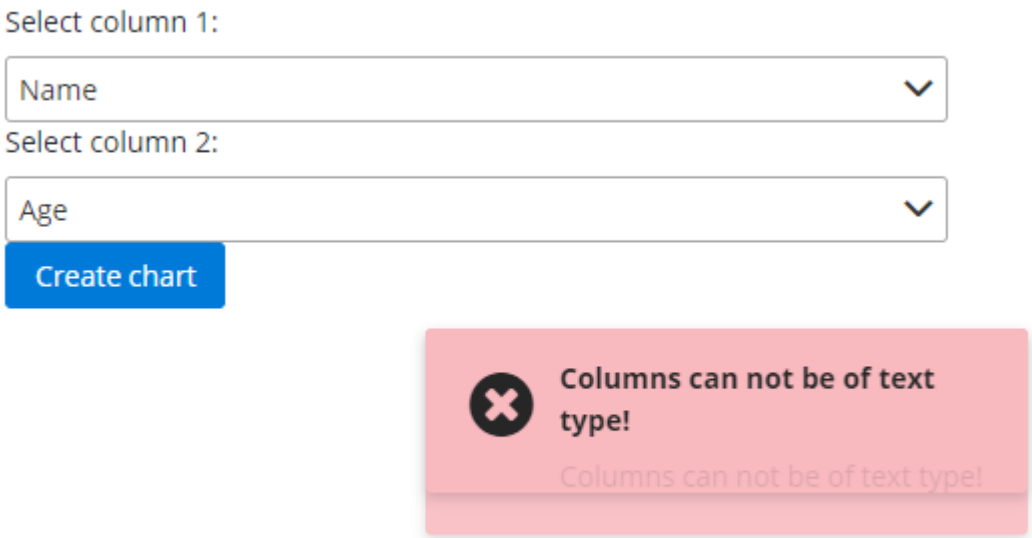


Рис. 4.15 Помилка побудови графіку

ВИСНОВКИ ДО РОЗДІЛУ 4

У даному розділі продемонстровано процес розгортання системи та проведено огляд інтерфейсу користувача.

ВИСНОВКИ

У ході виконання дипломного проекту було проведено аналіз процесів формування звітів та вже існуючих автоматизованих рішень.

Було спроектовано та розроблено сервіс що надає можливість створення, зберігання та доповнення різноманітних наборів даних, а також формування звітів на основі зібраної інформації. Розроблена система має ряд переваг, що не реалізовані у проаналізованих існуючих аналогах, а саме:

- можливость побудови цілісних звітів, що складаються з декількох наборів даних та містять різноманітні графіки та діаграми;
- можливість експорту готових наборів даних до системи;
- можливість імпорту даних у форматах .csv, .xls та .pdf.

Також було створено всю необхідну проектну документацію, схеми процесів системи, варіантів використань та описано керівництво для користувачів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Клієнт-серверна архітектура [Електронний ресурс]: (Стаття) / Вікіпедія Електрон. дан. (1 файл). – 2018. – Режим доступу: https://uk.wikipedia.org/wiki/Клієнт-серверна_архітектура – Назва з екрана.
2. MVC [Електронний ресурс]: (Стаття) / Вікіпедія. – Електрон. дан. (1 файл). – 2017. – Режим доступу: https://uk.wikipedia.org/wiki/Модель-вид-контролер – Назва з екрана.
3. Класифікація звітності та її користувачі [Електронний ресурс]: (Стаття) – Електрон. дан. (1 файл). – 2017. – Режим доступу: https://pidruchniki.com/1375040153277/buhgalterskiy_oblik_ta_audit/klasifikatsiya_zvitnosti_koristuvachi

ДОДАТОК А

Система формування звітів

Текст програми

ІАЛЦ.467200.007 А1

Листів 25

Київ – 2019

```
@Service
public class DatasetJsonService {

    @Autowired
    private DatasetJsonRepository datasetJsonRepository;

    public DatasetJson getByDatasetId(Long datasetId) {
        return datasetJsonRepository.getDatasetJsonByDatasetId(datasetId);
    }

    public DatasetJson saveDataset(DatasetJson datasetJson) {
        return datasetJsonRepository.save(datasetJson);
    }
}

@Service
public class DatasetService {

    @Autowired
    private DatasetRepository datasetRepository;

    public Dataset findById(Long id) {
        return datasetRepository.findById(id).get();
    }
}

@Service
public class UserService {

    private final UserRepository userRepository;
    private final PasswordEncoder passwordEncoder;

    @Autowired
    public UserService(UserRepository userRepository, PasswordEncoder
passwordEncoder) {
        this.userRepository = userRepository;
        this.passwordEncoder = passwordEncoder;
    }

    public User getCurrentUser() {
        org.springframework.security.core.userdetails.User user =
                (org.springframework.security.core.userdetails.User)
SecurityContextHolder.getContext().getAuthentication().getPrincipal();
        return findByUsername(user.getUsername());

    }

    public User findByUsername(String username) {
        return userRepository.findByUsername(username);
    }

    public void saveUser(User user) {
        user.setPassword(passwordEncoder.encode(user.getPassword()));
        user.setMatchingPassword(user.getPassword());
        user.setRoles(new
HashSet<>(Collections.singletonList(Role.USER)));
        user.setActive(true);
        userRepository.save(user);
    }
}
```

```
public interface DatasetJsonRepository extends
MongoRepository<DatasetJson, Long> {

    DatasetJson getDatasetJsonByDatasetId(Long datasetId);
}

public interface DatasetRepository extends JpaRepository<Dataset, Long> {

    List<Dataset> findAllByUserAndNameContainingIgnoreCase(User user,
String name);
}

public interface UserRepository extends JpaRepository<User, Long> {

    User findByUsername(String username);
}

@PasswordMatches
@Entity
@Table(name = "USER")
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    @Length(min = 5, message = "Your username must have at least 5
characters!")
    private String username;

    @Length(min = 6, message = "Your password must have at least 6
characters!")
    private String password;

    @Transient
    private String matchingPassword;

    @Email(message = "Provide a valid e-mail")
    @Column(name = "e_mail")
    private String email;

    private boolean active;

    @ElementCollection(targetClass = Role.class, fetch = FetchType.EAGER)
    @CollectionTable(name = "role", joinColumns = @JoinColumn(name =
"user_id"))
    @Enumerated(EnumType.STRING)
    @Column(name = "role")
    private Set<Role> roles;

    public User() {
    }

    public User(long id, String username, String password, String email) {
        this.id = id;
        this.username = username;
        this.password = password;
        this.email = email;
    }

    public Long getId() {
        return id;
    }
```

```
    public void setId(Long id) {
        this.id = id;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getMatchingPassword() {
        return matchingPassword;
    }

    public void setMatchingPassword(String matchingPassword) {
        this.matchingPassword = matchingPassword;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public boolean isActive() {
        return active;
    }

    public void setActive(boolean active) {
        this.active = active;
    }

    public Set<Role> getRoles() {
        return roles;
    }

    public void setRoles(Set<Role> roles) {
        this.roles = roles;
    }
}

public enum Role {
    USER, ADMIN
}
```

```
public enum ColumnTypeEnum {

    TEXT_TYPE("Text"),
    NUMBER_TYPE("Number"),
    DOUBLE_TYPE("Float number");

    private String name;

    ColumnTypeEnum(String type) {
        this.name = type;
    }

    public String getName() {
        return name;
    }
}

@Entity
@Table(name = "DATASET")
public class Dataset implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    @Column(name = "NAME")
    private String name;

    @ManyToOne
    @JoinColumn(name = "USER_ID")
    private User user;

    @OneToMany(mappedBy = "dataset", fetch = FetchType.EAGER, cascade =
CascadeType.ALL)
    private List<DatasetColumn> columns;

    public Dataset() {
    }

    public Dataset(String name, User user) {
        this.name = name;
        this.user = user;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public User getUser() {
        return user;
    }
```

```
    public void setUser(User user) {
        this.user = user;
    }

    public List<DatasetColumn> getColumns() {
        return columns;
    }

    public void setColumns(List<DatasetColumn> columns) {
        this.columns = columns;
    }
}
@Entity
@Table(name = "DATASET_COLUMN")
public class DatasetColumn implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    @Column(name = "NAME")
    private String name;

    @Enumerated(EnumType.STRING)
    @Column(name = "TYPE")
    private ColumnTypeEnum type;

    @ManyToOne(fetch = FetchType.LAZY, cascade = CascadeType.ALL)
    @JoinColumn(name = "dataset_id")
    private Dataset dataset;

    @Transient
    private long rowKey;

    public DatasetColumn() {
    }

    public DatasetColumn(String name, ColumnTypeEnum type, Dataset
dataset, long rowKey) {
        this.name = name;
        this.type = type;
        this.dataset = dataset;
        this.rowKey = rowKey;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public ColumnTypeEnum getType() {
```

```
        return type;
    }

    public void setType(ColumnTypeEnum type) {
        this.type = type;
    }

    public Dataset getDataset() {
        return dataset;
    }

    public void setDataset(Dataset dataset) {
        this.dataset = dataset;
    }

    public long getRowKey() {
        return rowKey;
    }

    public void setRowKey(long rowKey) {
        this.rowKey = rowKey;
    }
}

@Document(collection = "datasets")
public class DatasetJson {

    @Id
    private ObjectId id;

    private Long datasetId;

    private List<Long> columns;

    private List<List<String>> values;

    public Long getDatasetId() {
        return datasetId;
    }

    public void setDatasetId(Long datasetId) {
        this.datasetId = datasetId;
    }

    public List<Long> getColumns() {
        return columns;
    }

    public void setColumns(List<Long> columns) {
        this.columns = columns;
    }

    public List<List<String>> getValues() {
        return values;
    }

    public void setValues(List<List<String>> values) {
        this.values = values;
    }
}
```

```
public class DatasetJsonMapper {

    public static DatasetJson mapToJson(Dataset dataset,
List<List<String>> values) {
        DatasetJson datasetJson = new DatasetJson();
        datasetJson.setDatasetId(dataset.getId());

        List<Long> columns = dataset.getColumns()
                .stream()
                .map(DatasetColumn::getId)
                .collect(Collectors.toList());
        datasetJson.setColumns(columns);

        datasetJson.setValues(values);

        return datasetJson;
    }
}

@Scope("view")
@Controller("createDataset")
public class CreateDatasetBean {

    @Autowired
    private UserService userService;

    @Autowired
    private DatasetRepository datasetRepository;

    @Autowired
    private DatasetJsonService datasetJsonService;

    private Dataset dataset = new Dataset();
    private String datasetName;
    private DatasetColumn selectedColumn;
    private List<DatasetColumn> datasetColumns = new ArrayList<>();
    private long nextRowKey = 0;

    @PostConstruct
    public void init() {
        onAddNew();
    }

    public void deleteColumn() {
        datasetColumns.remove(selectedColumn);
    }

    public void onAddNew() {
        DatasetColumn column = new DatasetColumn();
        column.setDataset(dataset);
        column.setRowKey(nextRowKey);
        nextRowKey++;
        datasetColumns.add(column);
    }

    public boolean isEmptyColumns() {
        return datasetColumns.isEmpty();
    }
```

```
    public void saveDataset() {
        dataset.setUser(userService.getCurrentUser());
        dataset.setColumns(datasetColumns);
        dataset.setName(datasetName);
        dataset = datasetRepository.save(dataset);

datasetJsonService.saveDataset(DatasetJsonMapper.mapToJson(dataset,
Collections.emptyList()));

        clearParameters();

        FacesContext.getCurrentInstance()
                .addMessage("Successful", new FacesMessage("Dataset is
successfully saved"));
    }

    private void clearParameters() {
        dataset = new Dataset();
        datasetColumns = new ArrayList<>();
        datasetName = null;
        onAddNew();
    }

    public Dataset getDataset() {
        return dataset;
    }

    public void setDataset(Dataset dataset) {
        this.dataset = dataset;
    }

    public String getDatasetName() {
        return datasetName;
    }

    public void setDatasetName(String datasetName) {
        this.datasetName = datasetName;
    }

    public List<ColumnTypeEnum> getAvailableColumnTypes() {
        return Arrays.asList(ColumnTypeEnum.values());
    }

    public List<DatasetColumn> getDatasetColumns() {
        return datasetColumns;
    }

    public void setDatasetColumns(List<DatasetColumn> datasetColumns) {
        this.datasetColumns = datasetColumns;
    }

    public DatasetColumn getSelectedColumn() {
        return selectedColumn;
    }

    public void setSelectedColumn(DatasetColumn selectedColumn) {
        this.selectedColumn = selectedColumn;
    }

    public void setUserService(UserService userService) {
        this.userService = userService;
    }
```

```
    public void setDatasetRepository(DatasetRepository datasetRepository)
{
        this.datasetRepository = datasetRepository;
    }
}
@Scope("view")
@Controller("editDataset")
public class EditDatasetBean {

    @Autowired
    private UserService userService;

    @Autowired
    private DatasetService datasetService;

    @Autowired
    private DatasetJsonService datasetJsonService;

    private Dataset dataset;

    private DatasetJson datasetJson;

    private List<RowWrapper> rowWrappers = new ArrayList<>();

    private RowWrapper selectedRow;

    private long nextRowKey = 0;

    @PostConstruct
    public void init() {
        Long datasetId = Long.valueOf(

FacesContext.getCurrentInstance().getExternalContext().getRequestParameter
Map().get("datasetId"));
        dataset = datasetService.findById(datasetId);

        datasetJson = datasetJsonService.getByDatasetId(datasetId);

        List<List<String>> datasetJsonValues = datasetJson.getValues();
        for (List<String> valueList: datasetJsonValues) {
            int i = 0;
            Map<String, String> pairs = new HashMap<>();
            for (String value : valueList) {
                pairs.put(dataset.getColumns().get(i).getName(), value);
                i++;
            }
            rowWrappers.add(new RowWrapper(pairs));
        }

        if (rowWrappers.isEmpty()) {
            rowWrappers.add(new RowWrapper(new HashMap<>()));
        }
    }
```

```
public void saveChanges() {
        List<List<String>> rows = new ArrayList<>();
        for (RowWrapper rowWrapper: rowWrappers) {
            List<String> row = new ArrayList<>();
            for (DatasetColumn column : dataset.getColumns()) {
                row.add(rowWrapper.getValues().get(column.getName()));
            }
            rows.add(row);
        }
        datasetJson.setValues(rows);
        datasetJson = datasetJsonService.saveDataset(datasetJson);
    }

    public void deleteRow() {
        rowWrappers.remove(selectedRow);
    }

    public void onAddNew() {
        rowWrappers.add(new RowWrapper(new HashMap<>()));
    }

    public List<RowWrapper> getRowWrappers() {
        return rowWrappers;
    }

    public void setRowWrappers(List<RowWrapper> rowWrappers) {
        this.rowWrappers = rowWrappers;
    }

    public Dataset getDataset() {
        return dataset;
    }

    public void setDataset(Dataset dataset) {
        this.dataset = dataset;
    }

    public RowWrapper getSelectedRow() {
        return selectedRow;
    }

    public void setSelectedRow(RowWrapper selectedRow) {
        this.selectedRow = selectedRow;
    }

    public class RowWrapper {
        private long key;
        Map<String, String> values;

        public RowWrapper(Map<String, String> values) {
            this.key = nextRowKey;
            nextRowKey++;
            this.values = values;
        }

        public Map<String, String> getValues() {
            return values;
        }

        public void setValues(Map<String, String> values) {
            this.values = values;
        }
```

```
        public long getKey() {
            return key;
        }

        public void setKey(long key) {
            this.key = key;
        }
    }
}
@Scope("session")
@Controller("searchDataset")
public class SearchDatasetBean {

    @Autowired
    private UserService userService;

    @Autowired
    private DatasetRepository datasetRepository;

    private String datasetName;

    private List<Dataset> resultDatasets = new ArrayList<>();

    public void search() {
        resultDatasets =
datasetRepository.findAllByUserAndNameContainingIgnoreCase(userService.get
CurrentUser(), datasetName);
    }

    public boolean isEmptyResult() {
        return resultDatasets.isEmpty();
    }

    public List<Dataset> getResultDatasets() {
        return resultDatasets;
    }

    public void setResultDatasets(List<Dataset> resultDatasets) {
        this.resultDatasets = resultDatasets;
    }

    public String getDatasetName() {
        return datasetName;
    }

    public void setDatasetName(String datasetName) {
        this.datasetName = datasetName;
    }

    public void setUserService(UserService userService) {
        this.userService = userService;
    }

    public void setDatasetRepository(DatasetRepository datasetRepository)
{
        this.datasetRepository = datasetRepository;
    }
}
```

```
package com.springboot.controller.beans.charts;
```

```
import com.springboot.model.dataset.*;
import com.springboot.service.DatasetJsonService;
import com.springboot.service.DatasetService;
import com.springboot.service.UserService;
import javafx.util.Pair;
import org.primefaces.model.chart.AxisType;
import org.primefaces.model.chart.LineChartModel;
import org.primefaces.model.chart.LineChartSeries;
import org.primefaces.model.chart.PieChartModel;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import javax.faces.application.FacesMessage;
import javax.faces.context.FacesContext;
import java.util.*;
import java.util.stream.Collectors;

@SuppressWarnings("Duplicates")
@Scope("view")
@Controller("createChart")
public class CreateChartBean {

    @Autowired
    private UserService userService;

    @Autowired
    private DatasetService datasetService;

    @Autowired
    private DatasetJsonService datasetJsonService;

    private String datasetName;

    private Dataset selectedDataset;

    private String selectedDatasetName;

    private List<Dataset> resultDatasets = new ArrayList<>();

    private ChartTypeEnum chartType;

    private PieChartModel pieModel;
    private String column1;
    private String column2;
    private ComputeTypeEnum computeType;

    private LineChartModel lineChart;

    private DatasetJson datasetJson;

    public void generateChart() {
        switch (chartType) {
            case PIE:
                createPieChart();
            case LINE:
                createLineChart();
        }
    }

    private void createLineChart() {
        List<String> columnNames =
```

```
selectedDataset.getColumns().stream().map(DatasetColumn::getName).collect(Collectors.toList());
        List<ColumnTypeEnum> columnTypes =
selectedDataset.getColumns().stream().map(DatasetColumn::getType).collect(Collectors.toList());

        int c1 = columnNames.indexOf(column1);
        int c2 = columnNames.indexOf(column2);

        ColumnTypeEnum type1 = columnTypes.get(c1);
        ColumnTypeEnum type2 = columnTypes.get(c2);

        if (type1 == ColumnTypeEnum.TEXT_TYPE || type2 == ColumnTypeEnum.TEXT_TYPE) {
            FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_ERROR, "Columns can not
be of text type!", null);
            FacesContext.getCurrentInstance().addMessage(null, message);
            return;
        }


        List<List<String>> values = datasetJson.getValues();

        List<Pair<Double, Double>> data = new ArrayList<>();

        for (List<String> row : values) {
            data.add(new Pair<>(Double.valueOf(row.get(c1)), Double.valueOf(row.get(c2))));
        }


        lineChart = new LineChartModel();
        LineChartSeries series = new LineChartSeries();
        for (Pair<Double, Double> pair : data) {
            series.set(pair.getKey(), pair.getValue());
        }

        lineChart.getAxis(AxisType.X).setLabel(column1);
        lineChart.getAxis(AxisType.Y).setLabel(column2);
        lineChart.addSeries(series);
    }

    private void createPieChart() {
        List<String> columnNames =
selectedDataset.getColumns().stream().map(DatasetColumn::getName).collect(Collectors.toList());

        int c1 = columnNames.indexOf(column1);
        int c2 = columnNames.indexOf(column2);

        List<List<String>> values = datasetJson.getValues();

        Map<String, List<String>> pieMap = new HashMap<>();

        for (List<String> row : values) {
            pieMap.computeIfAbsent(row.get(c1), k -> new ArrayList<>());
            pieMap.get(row.get(c1)).add(row.get(c2));
        }

        Map<String, Number> map = new HashMap<>();

        for (Map.Entry<String, List<String>> entry : pieMap.entrySet()) {
            double sum = 0;
            for (String value : entry.getValue()) {
                sum += Double.valueOf(value);
            }
            map.put(entry.getKey(), sum);
```

```
        }

        pieModel = new PieChartModel();
        pieModel.setData(map);
        pieModel.setLegendPosition("w");

    }

    public void searchDataset() {
        clean();
        resultDatasets =
datasetService.findAllByUserAndNameContainingIgnoreCase(userService.getCurrentUser(), datasetName);
    }

    private void clean() {
        selectedDataset = null;
        chartType = null;
    }

    public boolean isLineChart() {
        return chartType == ChartTypeEnum.LINE;
    }

    public boolean isBarChart() {
        return chartType == ChartTypeEnum.BAR;
    }

    public boolean isPieChart() {
        return chartType == ChartTypeEnum.PIE;
    }

    public String getDatasetName() {
        return datasetName;
    }

    public void setDatasetName(String datasetName) {
        this.datasetName = datasetName;
    }

    public Dataset getSelectedDataset() {
        return selectedDataset;
    }

    public void setSelectedDataset(Dataset selectedDataset) {
        this.selectedDataset = selectedDataset;
    }

    public List<Dataset> getResultDatasets() {
        return resultDatasets;
    }

    public void setResultDatasets(List<Dataset> resultDatasets) {
        this.resultDatasets = resultDatasets;
    }

    public ChartTypeEnum getChartType() {
        return chartType;
    }

    public void setChartType(ChartTypeEnum chartType) {
        this.chartType = chartType;
    }
```

```
    public boolean getEmptyResult() {
        return resultDatasets.isEmpty();
    }

    public List<ChartTypeEnum> getAvailableChartTypes() {
        return Arrays.asList(ChartTypeEnum.values());
    }

    public List<ComputeTypeEnum> getAvailableComputeTypes() {
        return Arrays.asList(ComputeTypeEnum.values());
    }

    public boolean getDatasetIsSelected() {
        return selectedDataset != null;
    }

    public void allSelected() {
        selectedDataset = datasetService.findByUserAndName(userService.getCurrentUser(),
selectedDatasetName);
        datasetJson = datasetJsonService.getByDatasetId(selectedDataset.getId());
    }

    public String getSelectedDatasetName() {
        return selectedDatasetName;
    }

    public void setSelectedDatasetName(String selectedDatasetName) {
        this.selectedDatasetName = selectedDatasetName;
    }

    public PieChartModel getPieModel() {
        return pieModel;
    }

    public void setPieModel(PieChartModel pieModel) {
        this.pieModel = pieModel;
    }

    public String getColumn1() {
        return column1;
    }

    public void setColumn1(String column1) {
        this.column1 = column1;
    }

    public String getColumn2() {
        return column2;
    }

    public void setColumn2(String column2) {
        this.column2 = column2;
    }

    public ComputeTypeEnum getComputeType() {
        return computeType;
    }

    public void setComputeType(ComputeTypeEnum computeType) {
        this.computeType = computeType;
    }
```

```
    public LineChartModel getLineModel() {
        return lineChart;
    }

    public void setLineModel(LineChartModel lineChart) {
        this.lineChart = lineChart;
    }
}
```

```
package com.springboot.model.dataset;

public enum ChartTypeEnum {

    LINE("Line chart"), BAR("Bar chart"), PIE("Pie chart");

    private String name;

    ChartTypeEnum(String type) {
        this.name = type;
    }

    public String getName() {
        return name;
    }
}
```

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:p="http://primefaces.org/ui" xmlns:h="http://xmlns.jcp.org/jsf/html">
<h:head>
    <link rel="stylesheet" href="../static/css/bootstrap.min.css"/>

    <script type="text/javascript">
        //<![CDATA[
        function exportLineChart() {
            //export image
            $('#output').empty().append(PF('lineChart').exportAsImage());

            //show the dialog
            PF('dlg').show();
        }
        //]]>
    </script>

    <h:outputScript name="webjars/pdfmake/0.1.36/build/pdfmake.js" />
    <h:outputScript name="webjars/pdfmake/0.1.36/build/vfs_fonts.js" />

    <script type="text/javascript">
        //<![CDATA[
        function exportLineChartInPDF() {
            debugger;
            var docDefinition = {
                pageSize: 'A4',
                pageMargins: [ 40, 60, 40, 60 ],
                content: [
                    {
                        image: $(PF('lineChart').exportAsImage()).attr('src')
                    }
                ]
```

```
            };
            pdfMake.createPdf(docDefinition).open();
            pdfMake.createPdf(docDefinition).download('primefaces-charts.pdf');
        }
        function exportPieChartInPDF() {
            debugger;
            var docDefinition = {
                content: [
                    {
                        image: $(PF('pieChart').exportAsImage()).attr("src"),
                        width: 520.00,
                    }
                ]
            };
            pdfMake.createPdf(docDefinition).open();
            pdfMake.createPdf(docDefinition).download('primefaces-charts.pdf');
        }
        //]]>
    </script>


</h:head>

<h:body>
    <!--@elvariable id="createChart" type="com.springboot.controller.beans.charts.CreateChartBean"-->
    <h:panelGrid columns="2" cellspacing="4" cellpadding="4" style="vertical-align: top; height:
100%" columnClasses="align-top, align-top, align-top">
        <ui:include src="menu/leftMenu.xhtml"/>

        <h:panelGroup id="pnl_search_dataset">
            <h:form id="search_dataset_form">
                <h5>Create chart</h5>
                <p:growl/>
                <h:outputText value="Dataset name:" style="margin-right: 5px"/>
                <p:inputText id="datasetName" size="10" value="#{createChart.datasetName}"
required="true" requiredMessage="Provide dataset name!"/>
                <p:commandButton value="Search" ajax="false" update="pnl_search_results"
action="#{createChart.searchDataset}"/>
            </h:form>

            <h:form id="pnl_search_results" rendered="#{not createChart.emptyResult}">
                <p:outputLabel value="Select dataset:" />
                <p:selectOneMenu  value="#{createChart.selectedDatasetName}" style="width: 100%">
                    <f:selectItems value="#{createChart.resultDatasets}" var="dataset"
                                   itemLabel="#{dataset.name}" itemValue="${dataset.name}"/>
                </p:selectOneMenu>

                <p:outputLabel value="Select chart type:" />
                <p:selectOneMenu value="#{createChart.chartType}" style="width: 100%">
                    <f:selectItems value="#{createChart.availableChartTypes}" var="chartType"
                                   itemLabel="#{chartType.name}" itemValue="${chartType}"/>
                </p:selectOneMenu>
                <p:commandButton value="Select" ajax="false" update="pnl_line_chart, pnl_bar_chart,
pnl_pie_chart" action="#{createChart.allSelected}"/>
            </h:form>

            <h:panelGroup id="pnl_line_chart" rendered="#{createChart.lineChart}">
                <h:form >
                    <p:outputLabel value="Select column 1:" />
                    <p:selectOneMenu  value="#{createChart.column1}" style="width: 100%">
                        <f:selectItems value="#{createChart.selectedDataset.columns}" var="column"
                                       itemLabel="#{column.name}" itemValue="${column.name}"/>
```

```
            </p:selectOneMenu>

            <p:outputLabel value="Select column 2:" />
            <p:selectOneMenu  value="#{createChart.column2}" style="width: 100%">
               <f:selectItems value="#{createChart.selectedDataset.columns}" var="column"
                              itemLabel="#{column.name}" itemValue="${column.name}"/>
            </p:selectOneMenu>
            <p:commandButton value="Create chart" ajax="false"
action="#{createChart.generateChart}"/>
         </h:form>
         <p:chart type="line" model="#{createChart.lineModel}" style="width:500px;height:300px"
rendered="#{createChart.lineModel != null}"  widgetVar="lineChart" />
         <p:commandButton type="button" value="Export" rendered="#{createChart.lineModel !=
null}" icon="pi pi-home" onclick="exportLineChartInPDF()"/>

         <p:dialog widgetVar="dlg" showEffect="fade" modal="true" header="Chart as an Image"
resizable="true">
            <p:outputPanel id="output" layout="block" style="width:800px;height:800px"/>
         </p:dialog>
      </h:panelGroup>

      <h:panelGroup id="pnl_bar_chart" rendered="#{createChart.barChart}">
         bar
      </h:panelGroup>

      <h:panelGroup id="pnl_pie_chart" rendered="#{createChart.pieChart}">
         <h:form >
            <p:outputLabel value="Select column 1:" />
            <p:selectOneMenu  value="#{createChart.column1}" style="width: 100%">
               <f:selectItems value="#{createChart.selectedDataset.columns}" var="column"
                              itemLabel="#{column.name}" itemValue="${column.name}"/>
            </p:selectOneMenu>

            <p:outputLabel value="Select column 2:" />
            <p:selectOneMenu  value="#{createChart.column2}" style="width: 100%">
               <f:selectItems value="#{createChart.selectedDataset.columns}" var="column"
                              itemLabel="#{column.name}" itemValue="${column.name}"/>
            </p:selectOneMenu>
            <p:commandButton value="Create chart" ajax="false"
action="#{createChart.generateChart}"/>
         </h:form>
         <p:chart type="pie" model="#{createChart.pieModel}" rendered="#{createChart.pieModel !=
null}" style="width: 500px; height: 300px" widgetVar="pieChart"/>
         <p:commandButton type="button" value="Export" rendered="#{createChart.pieModel !=
null}" icon="pi pi-home" onclick="exportPieChartInPDF()"/>
      </h:panelGroup>

   </h:panelGroup>

</h:panelGrid>

</h:body>

</html>

<ui:composition xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets" xmlns:p="http://primefaces.org/ui"
      xmlns:h="http://java.sun.com/jsf/html">
   <h:panelGroup id="pnl_menu" style="float: left; height: 100%; margin-right: 10px" >
      <h:form>
         <p:growl id="messages" showDetail="true" />
```

```
        <p:tieredMenu style="width:200px">
          <p:submenu label="Datasets">
            <p:menuitem value="Create new" url="/xhtml/createDataset.xhtml"/>
            <p:menuitem value="Search" url="/xhtml/searchDataset.xhtml"/>
          </p:submenu>
          <p:submenu label="Charts">
            <p:menuitem value="Create new" url="/xhtml/createChart.xhtml"/>
          </p:submenu>
          <p:separator />
          <p:menuitem value="Sign out" action="#{menuView.delete}" update="messages"
ajax="false"/>

        </p:tieredMenu>

        <p:separator />
      </h:form>
    </h:panelGroup>
</ui:composition>



<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:p="http://primefaces.org/ui" xmlns:a4j="http://java.sun.com/jsf/html">
<h:head>
    <link rel="stylesheet" href="../static/css/bootstrap.min.css"/>
</h:head>

<h:body>
    <!--@elvariable id="createDataset" type="com.springboot.controller.beans.dataset.CreateDatasetBean"--
>
    <h:panelGrid columns="3" cellspacing="4" cellpadding="4" style="vertical-align: top; height:
100%" columnClasses="align-top, align-top, align-top">
        <ui:include src="menu/leftMenu.xhtml"/>
        <h:panelGroup>

            <h:panelGroup id="pnl_construct_dataset" style="vertical-align: top; width: 300px">
                <h:form id="add_columns_form" style="vertical-align: top">
                    <h5>Construct dataset</h5>

                    <p:growl id="msgs"/>
                        <p:dataTable id="columnsTable" widgetVar="columnsTable" var="column"
value="#{createDataset.datasetColumns}" rowKey="#{column.rowKey}"
                                     editable="true" selection="#{createDataset.selectedColumn}"
selectionMode="single" >

                            <p:column headerText="Column name">
                                <p:cellEditor>
                                    <f:facet name="output">
                                        <h:outputText value="#{column.name}"/>
                                    </f:facet>
                                    <f:facet name="input">
                                        <h:inputText value="#{column.name}" style="width: 100%; height: 100%;"/>

                                    </f:facet>
                                </p:cellEditor>
                            </p:column>

                            <p:column headerText="Column type">
                                <p:cellEditor>
```

```
                        <f:facet name="output">
                            <h:outputText value="#{column.type.name}"/>
                        </f:facet>
                        <f:facet name="input">
                            <p:selectOneMenu id="select_type" value="#{column.type}" style="width:
100%">
                                <f:selectItems value="#{createDataset.availableColumnTypes}"
var="columnType"
                                               itemLabel="#{columnType.name}" itemValue="${columnType}"/>
                            </p:selectOneMenu>
                        </f:facet>
                    </p:cellEditor>
                </p:column>

                <p:column style="width:35px">
                    <p:rowEditor />
                </p:column>

            </p:dataTable>

            <p:contextMenu for="columnsTable">
                <p:menuitem value="Delete" update="columnsTable"
action="#{createDataset.deleteColumn}"/>
            </p:contextMenu>

            <div class="ui-g">
                <div class="ui-g-12">
                    <p:commandButton value="Add column" styleClass="ui-priority-primary"
process="@this" update=":add_columns_form:msgs"
                                     action="#{createDataset.onAddNew()}"
oncomplete="PF('columnsTable').addRow();"/>
                </div>
            </div>
        </h:form>
    </h:panelGroup>

    <h:panelGroup id="pnl_create_dataset">
        <h:form id="create_dataset_form">
            <p:growl/>
            <p:outputLabel for="datasetName" value="Dataset name: " style="margin-right: 5px"/>
            <p:inputText id="datasetName" size="10" value="#{createDataset.datasetName}"
required="true" requiredMessage="Provide dataset name!" style="margin-right: 10px"/>
            <p:commandButton value="Save dataset" styleClass="ui-priority-secondary"
                             update="pnl_create_dataset pnl_construct_dataset"
disabled="#{createDataset.emptyColumns}"
                             action="#{createDataset.saveDataset}" ajax="false"/>
        </h:form>
    </h:panelGroup>
</h:panelGroup>


<!--<h:panelGroup>-->
    <!--<h4>Preview dataset</h4>-->
<!--</h:panelGroup>-->


</h:panelGrid>


</h:body>

</html>
```

```
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:ui="http://java.sun.com/jsf/facelets"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:p="http://primefaces.org/ui" xmlns:a4j="http://java.sun.com/jsf/html"
    xmlns:c="http://java.sun.com/jsp/jstl/core">
<h:head>
   <link rel="stylesheet" href="../static/css/bootstrap.min.css"/>
</h:head>

<h:body>
   <!--@elvariable id="editDataset" type="com.springboot.controller.beans.dataset.EditDatasetBean"-->
   <h:panelGrid columns="2" cellspacing="4" cellpadding="4" style="vertical-align: top; height:
100%" columnClasses="align-top, align-top, align-top">
      <ui:include src="menu/leftMenu.xhtml"/>

      <h:panelGroup id="pnl_edit_dataset" rendered="#{editDataset.dataset != null}">
         <h5 style="align-content: center">Edit #{editDataset.dataset.name}</h5>

        <h:form id="pnl_search_results">

           <h:commandLink>
              <p:graphicImage url="../static/img/excel.png" width="24"/>
              <p:dataExporter type="xls" target="dataset" fileName="#{editDataset.dataset.name}" />
           </h:commandLink>

           <h:commandLink>
              <p:graphicImage url="../static/img/pdf.png" width="24"/>
              <p:dataExporter type="pdf" target="dataset" fileName="#{editDataset.dataset.name}"/>
           </h:commandLink>

           <h:commandLink>
              <p:graphicImage url="../static/img/csv.png" width="24"/>
              <p:dataExporter type="csv" target="dataset" fileName="#{editDataset.dataset.name}" />
           </h:commandLink>

           <p:growl id="msgs" showDetail="true"/>

           <p:dataTable id="dataset" widgetVar="dataset" var="row"
value="#{editDataset.rowWrappers}" rowKey="#{row.key}" editable="true"
                        paginatorTemplate="{CurrentPageReport}  {FirstPageLink} {PreviousPageLink}
{PageLinks} {NextPageLink} {LastPageLink} {Exporters}"
                        paginator="true" rows="20"
                        selection="#{editDataset.selectedRow}" selectionMode="single">

              <p:ajax event="rowEdit" />
              <p:ajax event="rowEditCancel"/>

              <p:columns filterBy="#{editDataset.columnValue}" sortBy="#{editDataset.columnValue}"
var="column" value="#{editDataset.dataset.columns}"  columnIndexVar="colIndex">
                 <f:facet name="header">
                    <h:outputText value="#{column.name}" />
                 </f:facet>
                    <p:cellEditor binding="#{editDataset.columnParameter}">
                       <f:attribute name="columnToGetValue" value="#{row.values[column.name]}" />
                       <f:facet name="output">
                          <h:outputText value="#{row.values[column.name]}" />
                       </f:facet>
                       <f:facet name="input">
                          <h:inputText value="#{row.values[column.name]}" style="width: 100%; height:
100%;"/>
                       </f:facet>
```

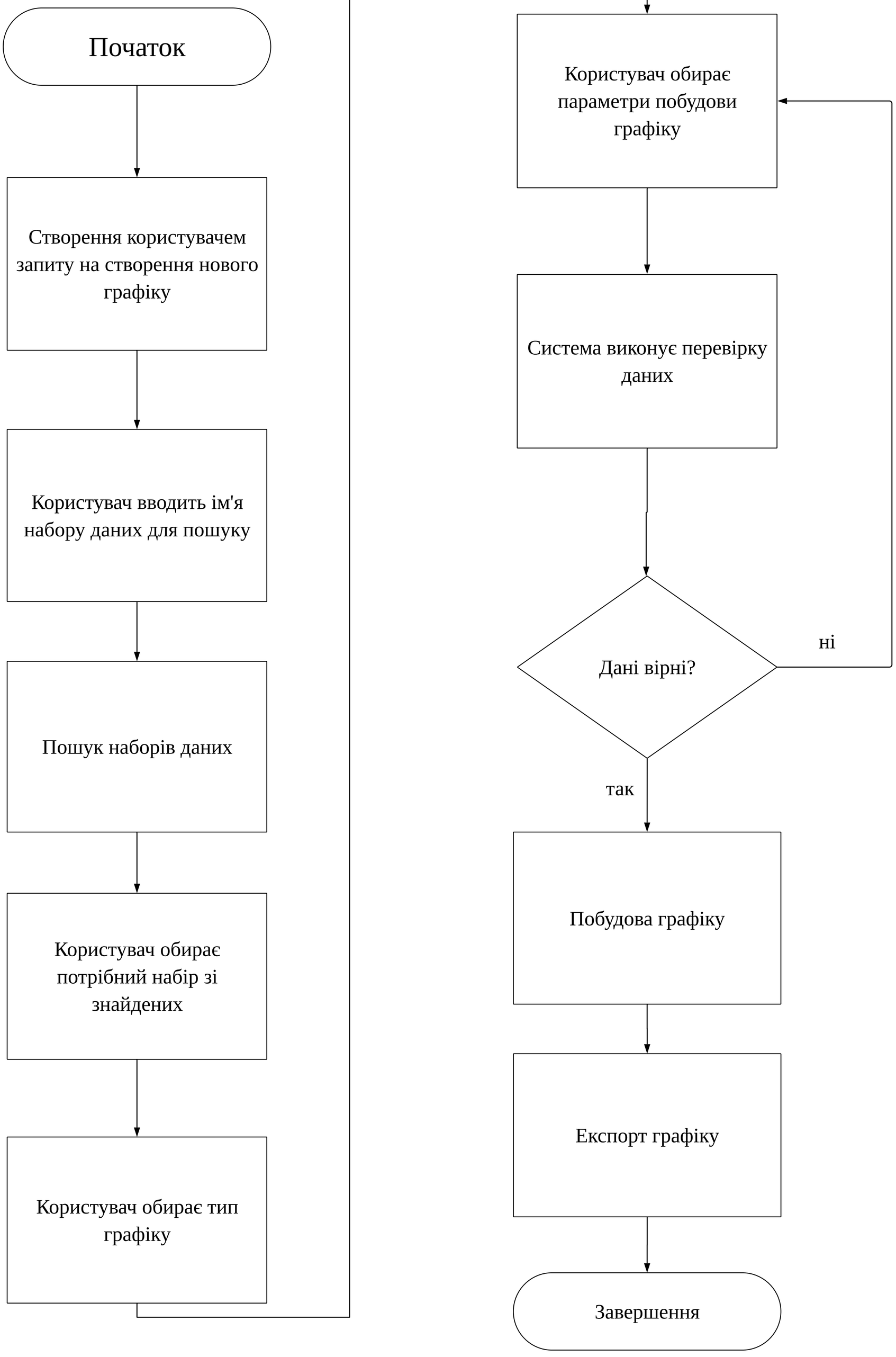
```
                </p:cellEditor>
            </p:columns>

            <p:column style="width:35px" exportable="false">
                <p:rowEditor />
            </p:column>
        </p:dataTable>

        <p:contextMenu for="dataset">
            <p:menuitem value="Delete" update="dataset" action="#{editDataset.deleteRow}"/>
        </p:contextMenu>

        <div class="ui-g">
            <div class="ui-g-12">
                <p:commandButton value="Add new row" styleClass="ui-priority-primary"
process="@this" update=":pnl_search_results:msgs"
                                 action="#{editDataset.onAddNew()}" oncomplete="PF('dataset').addRow();"/>
            </div>
            <div class="ui-g-12">
                <p:commandButton value="Save changes" styleClass="ui-priority-primary"
action="#{editDataset.saveChanges}"/>
            </div>
        </div>

    </h:form>
</h:panelGroup>

</h:panelGrid>


</h:body>

</html>
```

```
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:ui="http://java.sun.com/jsf/facelets"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:p="http://primefaces.org/ui" xmlns:a4j="http://java.sun.com/jsf/html">
<h:head>
    <link rel="stylesheet" href="../static/css/bootstrap.min.css"/>
</h:head>

<h:body>
    <!--@elvariable id="searchDataset" type="com.springboot.controller.beans.dataset.SearchDatasetBean"--
>
    <h:panelGrid columns="2" cellspacing="4" cellpadding="4" style="vertical-align: top; height:
100%" columnClasses="align-top, align-top, align-top">
        <ui:include src="menu/leftMenu.xhtml"/>

        <h:panelGroup id="pnl_search_dataset">
            <h:form id="search_dataset_form">
                <h5>Search datasets</h5>
                <p:growl/>
                <h:outputText value="Dataset name:" style="margin-right: 5px"/>
                <p:inputText id="datasetName" size="10" value="#{searchDataset.datasetName}"
required="true" requiredMessage="Provide dataset name!"/>
                <p:commandButton value="Search" ajax="false" update="pnl_search_results"
action="#{searchDataset.search}"/>
            </h:form>
```

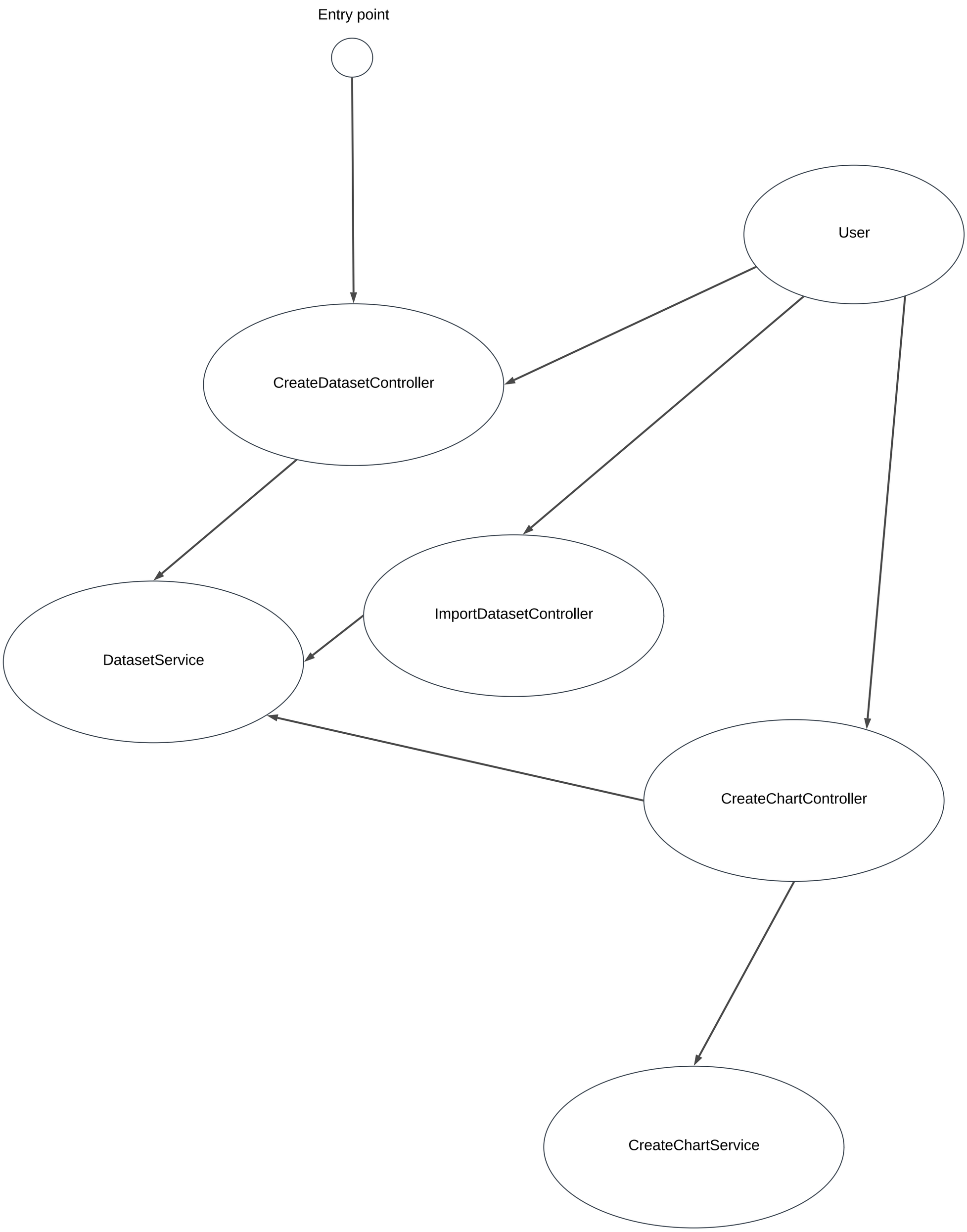
```
        <h:form id="pnl_search_results">
            <h5>Search result:</h5>
            <h:dataTable border="0" value="#{searchDataset.resultDatasets}" var="dataset"
rendered="#{not searchDataset.emptyResult}">
                <p:column>
                    <h:outputText value="#{dataset.name}" style="margin-right: 10px"/>
                    <p:linkButton value="Edit" outcome="editDataset.xhtml" style="margin-right: 10px">
                        <f:param name="datasetId" value="#{dataset.id}" />
                    </p:linkButton>
                    <p:commandButton value="Delete" ajax="false"
action="#{searchDataset.deleteDataset(dataset.id)}" />
                </p:column>
            </h:dataTable>
        </h:form>
    </h:panelGroup>

  </h:panelGrid>


</h:body>

</html>
```

```
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:ui="http://java.sun.com/jsf/facelets"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:p="http://primefaces.org/ui" xmlns:a4j="http://java.sun.com/jsf/html">
<h:head>
  <link rel="stylesheet" href="../static/css/bootstrap.min.css"/>
</h:head>

<h:body>
  <!--@elvariable id="searchDataset" type="com.springboot.controller.beans.dataset.SearchDatasetBean"--
>
  <h:panelGrid columns="2" cellspacing="4" cellpadding="4" style="vertical-align: top; height:
100%" columnClasses="align-top, align-top, align-top">
    <ui:include src="menu/leftMenu.xhtml"/>

    <h:panelGroup id="pnl_search_dataset">
        <h:form id="search_dataset_form">
            <h4>View datasets</h4>
            <p:growl/>
            <h:outputText value="Dataset name:" style="margin-right: 5px"/>
            <p:inputText id="datasetName" size="10" value="#{searchDataset.datasetName}"
required="true" requiredMessage="Provide dataset name!"/>
            <p:commandButton value="Search" ajax="false" update="pnl_search_results"
action="#{searchDataset.search}"/>
        </h:form>

        <h:form id="pnl_search_results">
            <h4>Search result:</h4>
            <p:dataTable  value="#{searchDataset.resultDatasets}" var="dataset">
                <p:column>
                    <h:outputText value="#{dataset.name}"/>
                    <p:linkButton value="View"  outcome="viewDataset.xhtml"/>
```

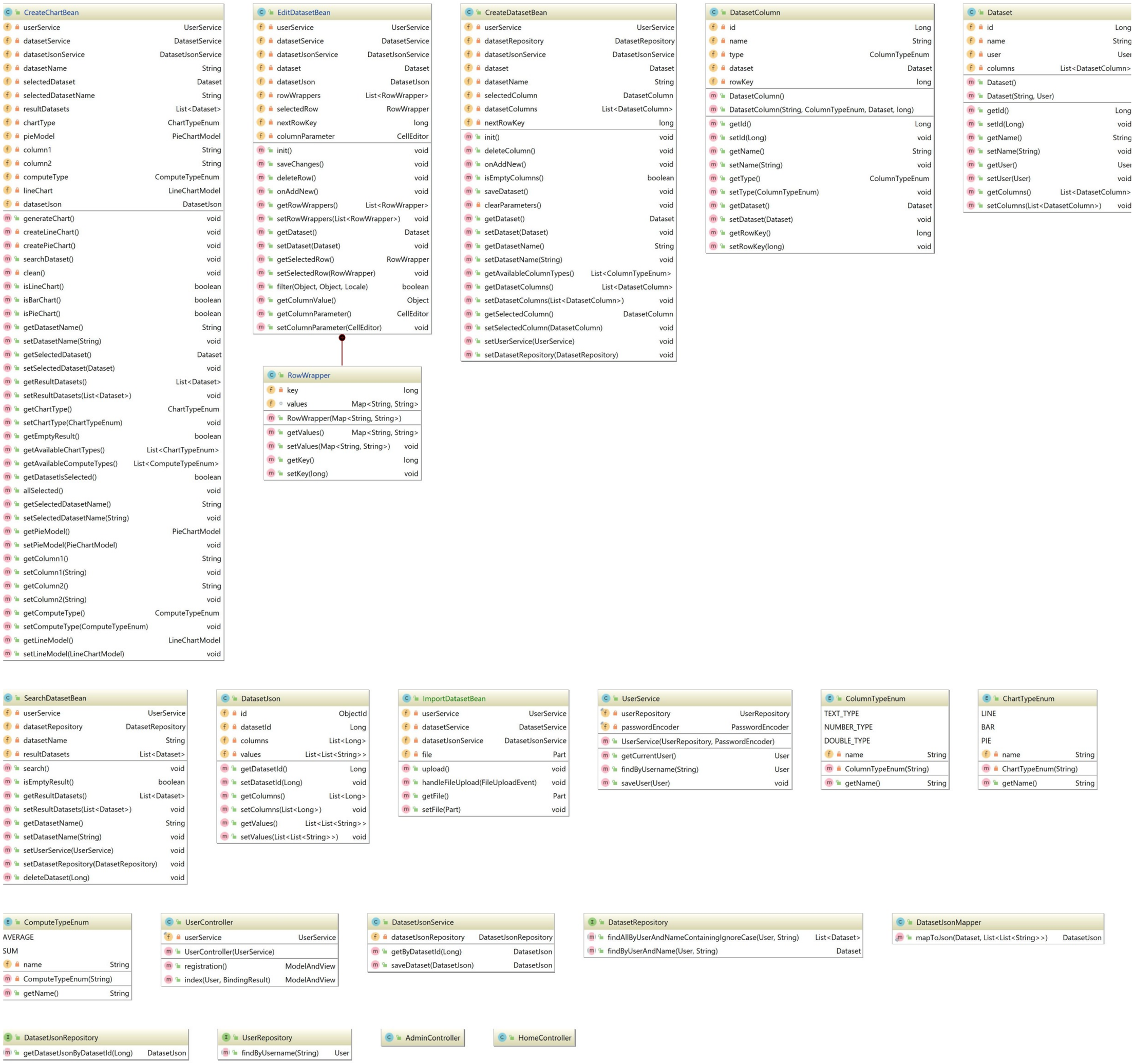
```
                    <p:linkButton value="Edit" outcome="editDataset.xhtml"/>
                </p:column>
            </p:dataTable>
        </h:form>
    </h:panelGroup>

</h:panelGrid>


</h:body>

</html>
```



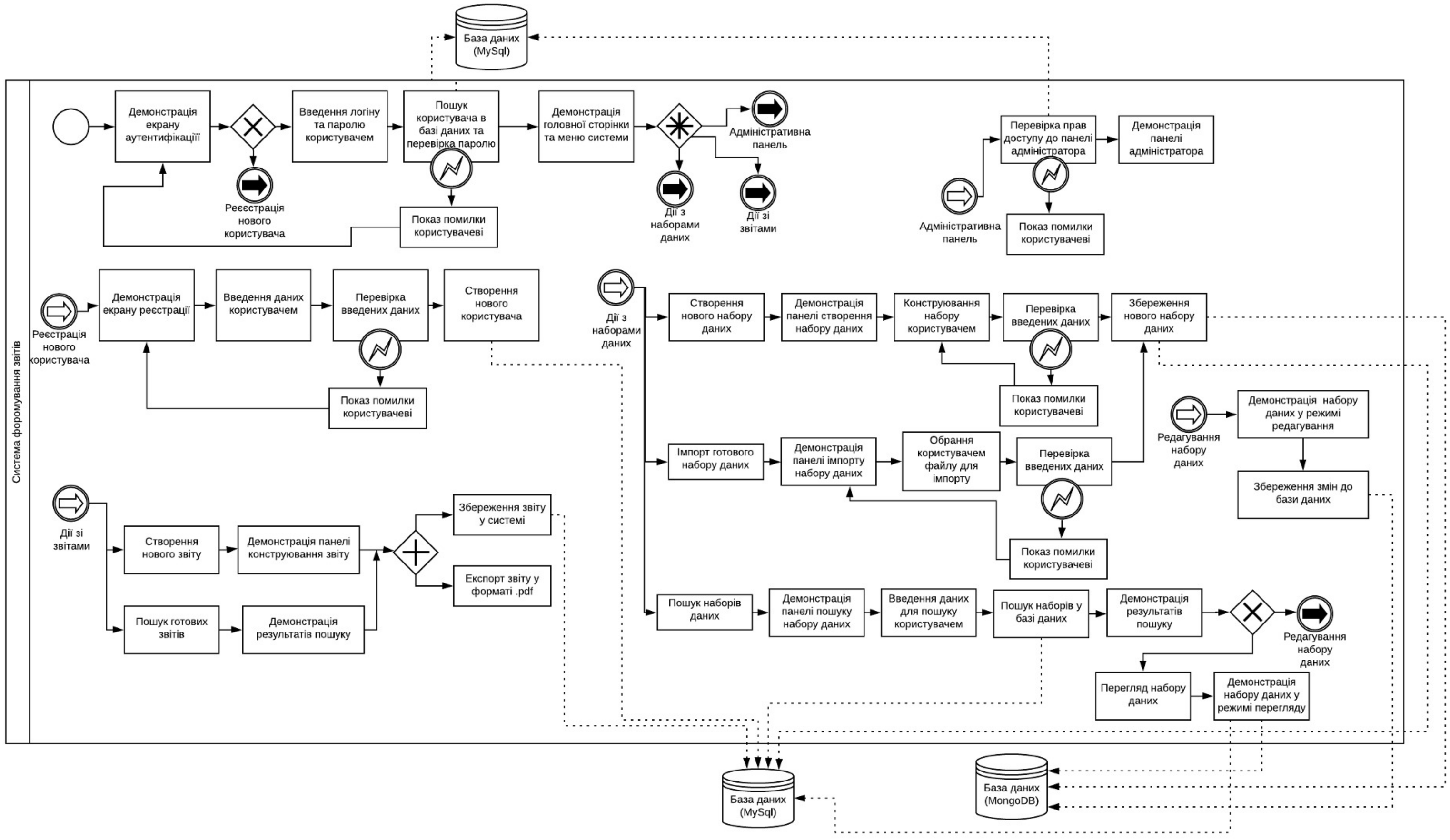
					ІАЛЦ.467200.004 Д1	
Змн.	Арк.	№ докум.	Підпис	Дата		
Розроб.		Романюк В. В.			Система формування звітів. Схема алгоритму роботи програми	Літ. / Арк. 1 / Аркушів 1
Перевір.		Регіда П. Г.				
Н. Контр.		Сімоненко В.П.				НТУУ „КПІ", ФІОТ, ІП-53
Затверд.		Стіренко С.Г.				



					ІАЛЦ.467200.005 Д2	
Змн.	Арк.	№ докум.	Підпис	Дата		
Розроб.		Романюк В. В.			Система формування звітів. Схема взаємодії класів системи	Літ. / Арк. 1 / Аркушів 1
Перевір.		Регіда П. Г.				
						НТУУ „КПІ", ФІОТ, ІП-53
Н. Контр.		Сімоненко В.П.				
Затверд.		Стіренко С.Г.				



					ІАЛЦ.467200.006 ДЗ	
Змн.	Арк.	№ докум.	Підпис	Дата		
Розроб.		Романюк В. В.			Система формування звітів. UML діаграма класів програми	Літ. / Арк. 1 / Аркушів 1
Перевір.		Регіда П. Г.				
						НТУУ „КПІ", ФІОТ, ІП-53
Н. Контр.		Сімоненко В.П.				
Затверд.		Стіренко С.Г.				



					ІАЛЦ.461000.007 Д4	
Змн.	Арк.	№ докум.	Підпис	Дата		
Розроб	.	Романюк В. В.			Система формування звітів. Схема бізнес процесів формування звіту	Літ. / Арк. 1 / Аркушів 1
Перевір.		Регіда П. Г.				
Н. Контр.		Сімоненко В.П.				НТУУ „КПІ", ФІОТ, ІП-53
Затверд.		Стіренко С.Г.				